HELSINKI SCHOOL OF ECONOMICS
Department of Business Technology

SOFTWARE AS A SERVICE:
OPPORTUNITIES AND CHALLENGES FOR SMALL SOFTWARE VENDORS

Information and Service Management
Master's thesis
Tapio Äijälä 73353
Spring 2008

Approved by the Head of the Department of Business Technology 12 / 5 2008, and awarded the grade good, 60p.

Petri Hallikainen Pirkko Lahdelma

ABSTRACT

Objectives

The goal of this master's thesis is to study how the SaaS model affects small- and medium-sized software vendors. What are the opportunities it brings? What are the challenges?

In this master's thesis I explain the different ways in which a small software vendor can start using the SaaS model and under which circumstances it has proven to be viable. I also discuss the different business and pricing models that can be used. In addition to this I cover the technological enablers and challenges that the SaaS concept brings, for instance in the fields of reliability and security of the service.

Methodology

The research was conducted as a literature study and an empirical case study. The first part of the study consists of an examination of academic articles, conference papers and other publications concerning the SaaS model. The second part consists of an empirical case study. The information was gathered by interviewing four small software vendors that develop and sell their own software products, either as delivered products, as services or as a mix these two different approaches.

Results

Probably the biggest finding of this study was to note that although there has not been much discussion on how the SaaS model affects especially small- and medium-sized software vendors the companies are already using the model very actively. This is proven by the fact that most of the successful large SaaS providers are companies that almost nobody knew about five years ago.

Customers buying software applications have become more price-sensitive and knowledgeable about different options available for them. Therefore a lot of consideration must be put on pricing models. Developing software solutions to be run over the internet creates of course new kind of problems and security threats. For a small software vendor it might be difficult to overcome customers' concerns regarding data security and ownership, loss and service's reliability.

Based on the results of the literature study and the case study it became evident that the SaaS model creates opportunities but also brings new challenges for small software vendors. Still, the opportunities definitely outnumber the challenges. Any small software vendor should at least assess the SaaS model and see how it would fit to her particular needs and situation.

Keywords: Software as a service, business models, pricing and sales strategies

TABLE OF CONTENTS

1 Introduction

1.1 A service-oriented view on software business

Software as a Service (SaaS) concept is a software application delivery model by which a software product vendor develops a software application, which is often web-based, and then hosts and operates the application over the internet for use by its customers (Sääksjärvi et al., 2005; Waters 2005). The SaaS model is a relatively new concept that has become increasingly popular during the last few years. It has many things in common with the so-called Application Service Provisioning (ASP) model and its origins can be traced back to very early times of computer industry and to the time-sharing services that were common back then (Kern et al., 2002; Walsh 2003).

The application service provisioning model became popular in the 1990s (Bennett, 2000). The biggest difference between the SaaS model and the ASP is that in the application service provisioning model the provider usually licenses a commercial software application, instead of developing the application by itself, and hosts it from centrally located servers (Luit Infotech, 2008). Sometimes the provider also customizes multiple versions of the application for different needs of its customers. However, the application service provisioning model was soon proven to be inefficient and much more costly than originally anticipated: Maintaining commercial software applications and customizing them was expensive, the application service provider did not have control over the future development of those applications and so forth. Many customers also soon discovered that the ASP providers lacked domain knowledge to understand their business well enough and customize applications to their needs.

Consumers have been using SaaS solutions already for several years without even being aware of it. Examples of this include such business-to-consumer services as eBay and PayPal (Nassil & Dasl & Shan, 2007). Among the business users the SaaS concept started to gain popularity at a time when IT executives became supremely fed up with the constantly growing costs of packaged software products (Levinson, 2007). These costs consist not only of money needed for the software licenses, but also money that must be spent on implementation, consultation, training and so forth.

The SaaS model offers significant benefits to both software vendors and their customers. For example, the customers do not pay for owning the software itself, but for using it as the pricing typically is either subscription or per-use -based. The model eliminates initial costs associated in buying software licenses and required hardware but also on-going costs and risks of installing, supporting and maintaining the software and the hardware.

One of the biggest advantages for software vendors is that the SaaS concept opens new markets for them. For instance, large established software developers can expand their market reach by providing their software applications as SaaS solutions to small and midsized companies. This way SaaS enables smaller customers to get access to "best-of-breed" applications that have earlier been out of their reach for example because of the high costs (Sääksjärvi et al., 2005). On the other hand, small software vendors can potentially expand rapidly outside their local markets or reach bigger clients than they could if they would develop and sell ordinary deployable software products. This is because often a new SaaS solution is not, at least in the beginning, actually sold to companies, but just to a group of users (SIIA, 2007). Therefore, a lucrative SaaS offering may turn highly successful inside a bigger corporation without the company's IT department's involvement or prior knowledge.

1.2 Motivation for the research

Some studies estimate that the market for SaaS solutions is growing as rapidly as 50% each year and some have even claimed that the traditional software is already dead (Choudhary, 2007b). Regardless of the growth rates really not being that high, it is not exaggeration to say that the SaaS model has had and will have a significant impact on the software business. This applies to software vendors of all sizes.

Although the SaaS concept has gained much attention both in trade press and scientific articles, there has not been much discussion on how the SaaS model affects especially small- and medium-sized software vendors. What are the opportunities it brings? What are the challenges? As the differences between the product and service business are considerable, the change of focus from more traditional business models to the SaaS model is not easily made (Lassila, 2006).

In this study I explain the different ways in which a small software vendor can start using the SaaS model and under which circumstances it has proven to be viable. I also discuss the different business and pricing models that can be used. In addition to this I cover the technological enablers and challenges that the SaaS concept brings, for instance in the fields of reliability and security of the service.

1.3 Research methodology

The research is conducted as a literature study and an empirical case study. The first part of the study consists of an examination of academic articles, conference papers and other publications concerning the SaaS model. The goal is to build a deep-enough understanding of the concept to analyze the opportunities and challenges of the SaaS model, especially from the perspective of small software vendors.

The second part consists of an empirical case study. The information was gathered by interviewing four small software vendors that develop and sell their own software products, either as delivered products, as services or as a mix these two different approaches. The objective is to understand how these companies see the SaaS model and the opportunities and challenges that it brings.

1.4 Research objectives and questions

The objectives of the literature study are:

- To study the key characteristics of software delivered as a service and the factors driving as well as the factors limiting the adoption of the SaaS model.
- To compare the differences between the SaaS model and traditional software business models and the effects on sales and pricing strategies.
- To identify the potential opportunities and the potential challenges of the SaaS model, especially from the perspective of small software vendors.

The objectives of the case study are:

- To analyze the case companies based on the findings from the literature. Do things really work in practice as described in the literature?

- To understand what have been the reasons why these companies have either decided to use or not to use the SaaS model in their business.
- To learn what kind of effects the SaaS model has had on their business, e.g. sales, profit and internationalization.
- To study the technical challenges and opportunities these companies have encountered when using the SaaS model.

1.5 Scope of the research

The research was done from the perspective of small software vendors, e.g. companies developing and selling software products. The main focus is to define the major opportunities and challenges of the SaaS model for small software vendors.

For the purpose of this study "small software vendor" means a company that has less than 25 employees. Probably most of this research can also be applied to larger companies but I decided to study the SaaS model particularly from the perspective of small software vendors as there is very little existing research available. In addition, small companies have often proven to be the true innovators in the software market. This has also been the case with modern web-based software solutions and the SaaS model. For large established software vendors, the SaaS market has not been appealing enough and as a result, they have tended to ignore it (Gartner, 2007). This has opened a window of opportunity for smaller software vendors.

Although some parts of the study may also naturally be applicable to the buyers of software applications and SaaS solutions, they have been left outside of the scope of this study. From the technical perspective the scope of the research is also limited to software services that are used by humans. In other words software services that are meant to be used by other software products or which can be used as building blocks when developing larger services are beyond the scope of this study.

1.6 The structure of the report

This report consists of five main chapters. The first chapter is an introduction to the software as a service model and the study as a whole. The motivation for the research,

the scope of the research and also the research methodology are covered in this section.

The second main chapter contains a literature study covering the SaaS model in more detail, including its definition and history. In addition some key characteristics of software delivered as a service are also defined. Based on the literature key factors both driving and limiting the adoption of the SaaS model are outlined. In the final part of this chapter there is a brief description of how the SaaS model affects business models and value chains.

The third main chapter takes a more detailed look at the SaaS model especially from the perspective of a small software vendor. Based on the earlier literature study I describe what kind of opportunities the new business models create and how these may affect sales and pricing strategies. Finally I discuss gaining and sustaining competitive advantage, technological enablers and also new technological challenges.

The fourth chapter of this report contains the empirical part. It consists of descriptions of the case companies and results of the interviews. At the end of the chapter there is analysis of the case study findings.

The last chapter of the report contains the discussion and conclusions. The major findings from both the literature study and the empirical part are combined together. Finally, some suggestions for further study are given.

2 Software as a service - the SaaS model

2.1 Definition and background

Hosted services, on-demand software, application service provisioning and many other similar terms are used to describe the same general concept of providing customers with software applications as a service. In the SaaS model the vendor of a software product installs and runs the application in its own data center while a customer still holds administrative control over the solution. As the application vendor is responsible for not only the software product itself, but provides also servers and other hosting equipment, maintenance and keeps the solution up-to-date, the customer is able to enjoy the benefits of the software application without the burden

of managing her own servers and taking care of installation and management of the product.

Although the term "software as a service" is relatively new, its roots can be traced back to earlier concepts and software delivery models sharing many similar ideas and goals. As Walsh puts it, even if technology may have changed dramatically over the past decades, its applications appear to have come a full circle (Walsh, 2003). To validate this argument he gives a definition from 1967 stating the following: "Time-sharing is a communications-oriented method of using computers. It is a technique that permits concurrent utilization of the same installation by two or more persons working at remote devices capable of direct, online access to the data processing equipment." One can replace "time-sharing" with "application service provisioning" or even with "software as a service" and the Ziegler's definition still holds true.

The ancestor of the SaaS model, called application service provisioning (ASP), became popular in the 1990s. It can be described as a form of selective outsourcing where a third-party organization rents generally available packaged software applications and related services (Bennett, 2000). In the purest form an application service provider is a company that is able to provide its customers with managed hardware, application software, network infrastructure and takes full responsibility of the whole solution for a monthly fee (Cope, 2000). In most cases application service providers did not develop applications themselves but took off-the-shelf applications, added some features such as web-based interfaces on top of the standard product, and ran the solution for their customers. Many of these providers did not have application development expertise or capabilities. It should be noted that because most of the applications were not developed to be hosted remotely, performance was often poor.

Many of the application service providers ran into severe problems quite soon (Levinson, 2007). While trying to serve the unique needs of each customer, they soon lost the economies of scale that were necessary for them to provide their services in a cost-effective manner. The SaaS model as a concept started to evolve from this problematic situation. The initial ideas of SaaS were first circulated in conferences and seminars in 2000 and 2001 and soon the SaaS model became a widely discussed topic in trade journals and reports. Some of the first reports and essays that explicitly

mentioned the SaaS include the 2004 IDC report by Amy Mizoras Konary (Konary, 2004) and Tim O'Reilly's essay "The Open Source Paradigm Shift" (O'Reilly, 2004).

In the spirit of Ziegler's definition of time-sharing computing mentioned above, Sääksjärvi et al. have presented a similar definition for the SaaS model:

> Software as a Service is time and location independent online access to a remotely managed server application, that permits concurrent utilization of the same application installation by a large number of independent users (customers), offers attractive payment logic compared to the customer value received, and makes a continuous flow of new and innovative software possible. (Sääksjärvi et al., 2005)

Lassila has defined the SaaS model as a networked e-commerce business model (Lassila, 2007). The focus is moving from owning the software to using the software. In short it can be said that the three most important differences between the SaaS model and the application service provisioning model are: 1) The SaaS model applies an e-commerce point-of-view instead of the application service provisioning model's outsourcing view, 2) The SaaS model emphasizes the capability and need to customize customer solutions and 3) The SaaS model is a coherent business model concerned with value creation and value appropriation whereas the application service provisioning model was more of a technical definition (Lassila, 2006).

At first the application service provisioning and later the SaaS model were mainly used in non-mission critical applications such as email, on-line backup and virus protection (Seltsikas & Currie, 2002). Today SaaS solutions are available for almost all business application areas including Customer Relationship Management (CRM), Human Resource Management (HRM) and so forth (Sun et al., 2007). One factor affecting availability has been the rapid development of web technologies as typical SaaS solutions are often developed specifically to be used online. With browser-based access, the learning curve for new, external applications is lower (Bennett, 2000).

2.2 Key characteristics of software delivered as a service

In many ways the SaaS model reminds us of the days of mainframe computing. Back then users connected to a central computer and accessed computer programs via their terminals. The programs were centrally controlled and available on demand

(Greschler & Mangan, 2002). In the SaaS concept the situation is very similar. In most cases so-called "dummy terminals" have just been replaced with users' desktop and laptop PCs and in some cases even with mobile phones or other handheld devices.

The SaaS model has been called the next paradigm in software application delivery and business. This type of online delivery of software applications has already been a goal for different parties for a long time (Dubey & Wagle, 2007). In the SaaS model the customer does not buy a license for a software application, install and manage it on her own servers but signs up for a software service developed and hosted by the vendor. Typically SaaS users pay a flat monthly or annual fee or a fee that is based on the usage of the service.

This type of outsourcing software applications over the internet can relieve the customer of heavy economic and technological pressures (McNabb, 2001). These include such pressures as the constant shortage of workforce with suitable IT skills and constant hardware and software investments required. The SaaS model also limits the required initial investments and costs (Sääksjärvi et al., 2005). From the customers' perspective the SaaS model can be seen as a way to reduce Total Cost of Ownership (TCO) and for software application vendors it opens up new payment models and revenue streams (Greschler & Mangan, 2002).

A software provider doing business using the SaaS model is able to achieve economies of scale in managing the required hardware, software and personnel resources as it can distribute these costs over many customers.

Instead of trying to provide everything to all customers, SaaS vendors typically provide single one-size-fits-all solutions (Levinson, 2007). All customers use the same version of the software and the underlying program code cannot be customized at all or only very little. Any features or new functionalities that are introduced in the product become available to all customers. This behavior is also-called multi-tenancy. It is one of the key competencies in achieving higher margin by leveraging economies of scale (Guo et al., 2007).

Reducing the total cost of ownership is not the only reason for customers in moving to the SaaS model. In many cases the more important reason is to cut the time of

implementation so that the new software solution can be taken into business usage as soon as possible (Nassil & Dasl & Shan, 2007; Sääksjärvi et al., 2005). The time between when an investment is made and when the benefits of a new software application are achieved is a critical component in making the investment decision (Waters, 2005). In some cases a long delay in implementation can even completely eliminate the Return On Investment (ROI) that the organization was expecting. In the worst case the business requirements of the organization may have changed so radically that the whole investment becomes obsolete.

According to Sääksjärvi et al. (Sääksjärvi et al., 2005) other widely accepted benefits of the SaaS model include for example the fact that SaaS makes it easier for customers to focus more on their core competencies as they do not need to worry about the IT infrastructure. Also upgrades are installed automatically enabling the customers to enjoy up-to-date technology (Walsh, 2003). In addition, from the perspective of small or medium size customers the SaaS model may open entirely new possibilities for them as they get access to "best-of-breed" software that would otherwise be too expensive for them to buy with the traditional annual or perpetual license model.

The SaaS model also makes it possible for the customers to access the software free of location and time (Sääksjärvi et al., 2005). This is a valuable option as work constantly becomes more mobile. Letting employees operating in the field to access company's information systems remotely, or allowing employees to work from home and other remote locations has a significant role in today's business environment.

As a negative effect the SaaS model may increase uncertainty as it distributes responsibilities of the solution among organizations (Walsh, 2003). For example, it is typical that the actual SaaS vendor has outsourced the hosting of the solution to an external party offering professional data center management. On the other hand, relying on a SaaS solution can provide small and midsize organizations with greater levels of security and reliability than they could achieve with their own organization (Walsh, 2003).

2.3 Factors driving the adoption of the SaaS model

The factors driving the adoption of the SaaS model can roughly be divided into two main categories: 1) Enablers of the model and problems of the traditional software engineering techniques 2) Delivery methods. Let us concentrate on the enablers first.

Since the early 1990s the internet technologies as a whole and especially technologies related to the World Wide Web have evolved significantly. Today the web can be seen as a platform that allows building even the most complex business applications on top of it. The breakthrough of the internet has increased the relative homogeny and ubiquity of workstations (Waters, 2005). Whether the desktop PC or the laptop of a business user is Windows, Macintosh or UNIX-based, it most probably is equipped with an internet connection and a web browser which has become the standard for universal graphical user interfaces (Tao, 2001).

Prior to the age of the internet standards, it was much more difficult to build generic applications that could be run everywhere despite of the underlying hardware and software setup. Today's web-based solutions are reliable enough for the most usage cases and the internet can be used to deliver business critical solutions to end users (Bennett & Timbrell, 2000). As the internet technologies have evolved, available network bandwidth has also increased drastically. Constantly growing data transfer speeds and improving quality of network connections have made it possible to remotely access data as easily and efficiently as it was stored in one's personal computer or in a local network. Today the physical location of data no longer matters as it is completely transparent both to applications and users (Waters, 2005).

It can be said that computing and software applications have in many ways turned into commodities instead of being something complex to obtain. As with all commodities, software application can simply be evaluated as cost centers and as such are suitable candidates for outsourcing and other cost reduction actions. Often when purchasing a software application and other IT solutions customers want to allow room for growth and tend to purchase more capacity than they actually need (Waters, 2005). Today's business organizations are usually under heavy pressure to make wise use of their capital expense budgets and thus purchasing more capacity than needed can be viewed as an inefficient use of capital.

In recent years IT outsourcing has evolved to provide more granularity than before (Tao, 2001). Instead of outsourcing their whole IT infrastructure companies can today choose to hand over as little as one application or a set of applications. Previously the only way to change a workflow of a software application was often to modify the program code. Today's software applications can typically be modified and customized in a variety of ways just by switching some parameters and modifying configuration files. The same base application can thus be used to provide multiple different solutions to customers.

Since the 1980s software business has evolved from custom-built solutions increasingly towards generic and off-the-shelf software products. Software solutions are rarely developed from scratch any more as there are suitable ready-made software products available for most industries and needs (Finkelstein & Kramer, 2000). Development is more focused on expanding preexisting IT systems, integrating different solutions together and interoperability.

The adoption of the SaaS model is also driven by the problems of the traditional software engineering techniques and delivery methods which are often said to be more supply than demand led (Bennett, 2001). A technology driven development technique might work well for development of embedded systems and other solutions with clear boundaries and a slow pace of evolvement, but for developing successful business applications more flexibility and speed is required.

Bennett states (Bennett, 2001) that for the past 40 years, the techniques, processes and methods of software development have been dominated by supply side issues. It can be said that the software application industry has been more oriented towards developers than customers. To achieve such levels of functionality, flexibility and time-to-market of changes which are demanded by the customers today, the software development industry has had to radically shift the way it operates. This change has led to a more demand-centric view and to a shift from the concern of whether a software solution will work towards how well it will work (Finkelstein & Kramer, 2000).

The change in the software development practices and methods has also affected how software solutions are deployed. Earlier taking a large software system into use might

have taken several months or even years which is often too long in today's fast paced business environment. In the SaaS model many of the typical implementation tasks are eliminated as the software solution is already up and running on the SaaS vendor's data center. Therefore deployment times usually are considerably shorter with the SaaS solutions than with software applications delivered using the traditional model.

The SaaS model also gives customers a possibility to gain immediate access to the latest software versions and updates. This is a very important factor as today's business and industrial processes are continually reorganized (Finkelstein & Kramer, 2000). As the customers' requirements change, the software solutions also must evolve. With the traditional model customers typically always need to wait for the next product version to be released to benefit from bug fixes and new features. In the SaaS model customers get access to those changes on an on-going basis. As soon as the software vendor makes an update to the software solution, it becomes available to the customers.

Just like the technology and software development and delivery methods have evolved, also the business models and ways of doing things have matured. Today both average software vendors and customers are more knowledgeable about the responsibilities, rights and obligations of both parties. This has made it easier for vendors and customers to enter into service agreements (Waters, 2005). Whereas earlier negotiations between a software vendor and a buyer might have taken months or even years and involved large number of people not only from the unit that actually will use the application, today SaaS solutions are sold very often using highly standardized agreement templates and pricing models. This gives customers the possibility to easily test drive different software applications and simply stop using those ones that do not suit the business needs or the requirements of the company. Before this has not been possible in most cases as it would have meant throwing already made investments away. For modern customers, who are constantly looking for new business areas it is a valuable option to be able to start and stop using a software solution very quickly whenever needed. Therefore it can be said the SaaS model matches very well the rapid and flexible business development needs of today's companies.

2.4 Factors limiting the adoption of the SaaS model

Although the SaaS model can be seen as a paradigm shift within the software business industry, there still are several factors that limit the adoption of the concept. Some types of business applications are better candidates for realizing the benefits of the SaaS model than others. In the following we will study some factors causing these differences one by one.

In overall it can be said that the SaaS model suits best non-strategic, non-mission-critical processes and functions (Levinson, 2007). Some organizations view their information systems as strategic assets and thus choose not to rely on applications provided using the SaaS model (Walsh, 2003). For example it is very natural that the customers have concerns about keeping their data in SaaS vendors' systems as they have no control over those systems (Levinson, 2007). This is for instance especially true for organizations in the medical and legal fields as they are typically very sensitive about their data. The loss of patient records or legal case documents could be devastating to a company or its clients (Walsh, 2003). In addition to the increased risk of losing business-critical data, potential risks for the customers include a SaaS solution performance related problems and the lack of enough possibilities of tailoring and integrating the solution to suit the specific needs (Lassila, 2007).

Similarly as many customers are reluctant to start utilizing SaaS solutions as they are afraid of the security of their data, many potential customers also question the reliability of software applications provided using the SaaS model. Although internet technologies have evolved significantly during the last ten years and even though the web has become a suitable platform most business application needs, it is not a fully problem-free solution. For instance many users still use their computers in environments with limited or no the internet connectivity at all, such as hotels, airports and so forth (Greschler & Mangan, 2002). Also business critical systems that must be available 24/7 without any interruptions do not tolerate any network outages and therefore are not suitable to be used remotely over the internet.

In case a software application needs to be tightly integrated to older legacy applications, a solution delivered using the SaaS model is often not the best possible choice. Earlier it was relatively typical that software applications were not developed

integration possibilities in mind and thus integrating a new SaaS solution with some old software application might require a large amount of additional programming and building new data exchange protocols and even requiring that the integration happens within the same operating environment (Waters, 2005). Also when a lot of customization work would be needed, sticking with a software application delivered using the traditional model often is a better choice (Levinson, 2007). Some SaaS vendors even refuse to do any customizations as that would drive up the costs and complexity and thus decreasing the benefits of the SaaS model.

As the SaaS market is still quite immature and most example cases are about delivery of non-critical applications, the potential customers find difficulties in evaluating the model (Desai et al., 2003). This induces a lack of confidence and trust which then negatively affects the sales. The lack of standards and best practices can be seen as a critical problem that affects customers' ability to make purchase decisions (Sun et al., 2007). This is especially true when more and more SaaS solutions become available from different vendors, through different service protocols and with various functionalities.

From a perspective of a new software entrepreneur the SaaS model may seem to offer endless opportunities, but it also poses major business challenges (Gardner, 2007). One of the largest challenges is caused by the pricing models used when selling SaaS solutions: A vendor typically gets paid monthly or quarterly, instead of receiving one large upfront payment for a perpetual software license. Even if the vendor would get all payments for the first year in advance, it would still receive significantly less cash than when selling a perpetual license using the traditional model. This means increased funding requirements for the company. The same cash-flow problems also affect existing software vendors if they aim to change their business models from the traditional model of selling annual or perpetual software licenses to the SaaS model. A company's revenues can initially drop even more significantly as the company does not receive any more license and consultation fees but a monthly service fee (Lassila, 2007). This situation of increased cash burn during the first years is presented in figure 1.

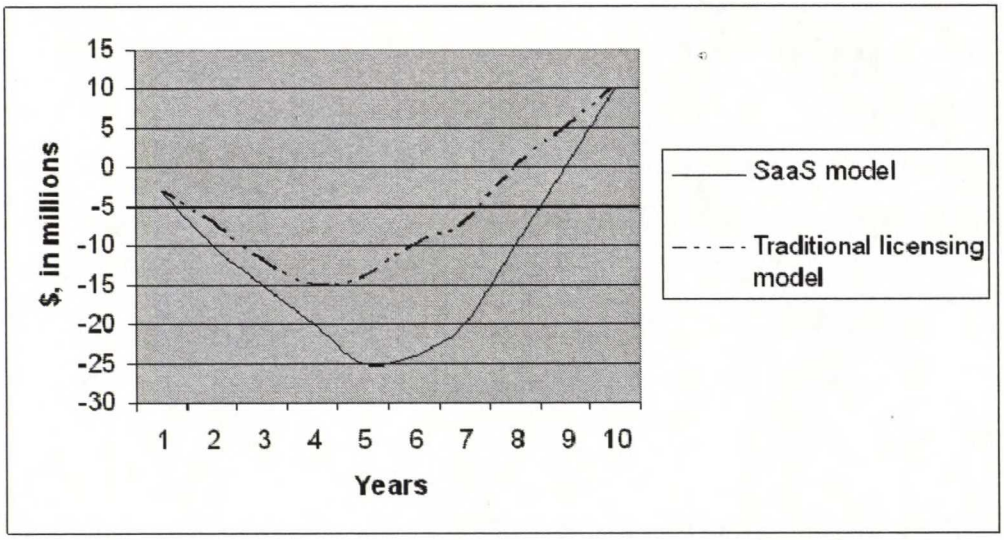

FIGURE 1: Example of cumulative cash burn for both SaaS and traditional licensing model. Source: Gardner, 2007

For a software vendor other potential risks include technical issues such as possible performance and scalability problems and also risks related to managing the partner network, for example. A company that has been developing its software product for long might not have the required experience and knowhow to run reliable and effective hosting operations. Also it might not have skills to develop a software platform that is scalable and can support a large number of simultaneous customers if the solution suddenly turns out to be very popular.

2.5 *Differences from traditional software business models*

The SaaS model and the traditional software licensing model, which often is based on a perpetual license, have several differences (Choudhary, 2007a). Previously almost all commercial software applications have been sold either on a basis of ownership or on a basis of perpetual right to use the application. This means that the customer has bought the object code, with some form of license to use it (Bennett, 2001). In this traditional model the customer has installed and managed the application on its own hardware (Waters, 2005). The customers have also optionally made additional payments for future upgrades of the software product. The SaaS model, on the other hand, is based on a subscription model.

In the traditional software licensing model software vendors' revenues generally come from three sources (Cusumano, 2007): License payments, maintenance fees and additional service fees. Customers make upfront license payments for the right to use a software application for example for one year or perpetually. In addition, they typically pay annual fixed service fees for a separate maintenance agreement. As long as the customers continue to pay the annual fee and do not breach the usage terms such as making the software available to more than the licensed number of users, they receive patches and updates to the product. Additional service fees are collected by the vendors from extra services that the customer requires, for instance from installing and integrating the application to the customer's environment, user training and so forth.

One of the biggest shortcomings of the traditional software business model are the unexpected costs. It has become common knowledge that the actual purchase cost forms often only a small part of the total cost of ownership of the software application (Waters, 2005). In his article Waters refers to a market study that has shown that up to 80% of IT budgets are not spent on buying hardware and software, but on system implementation and maintenance. This is shown using an iceberg metaphor in figure 2.

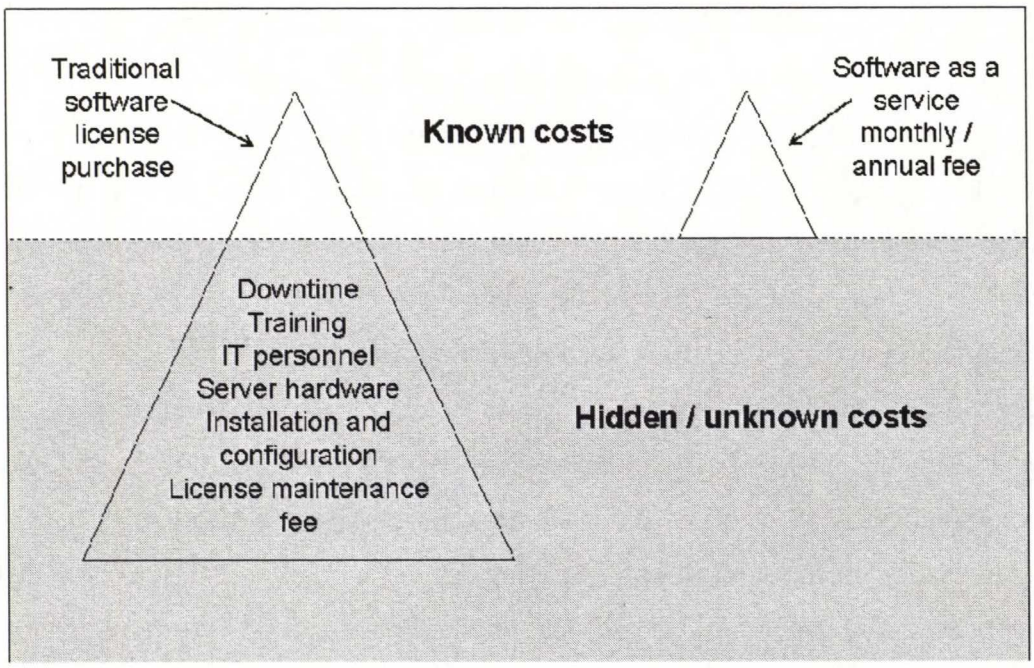

FIGURE 2: So-called iceberg metaphor of known and hidden costs occurring when purchasing software applications. Source: Adopted from Waters, 2005

As the SaaS model does not require large up-front investments from customers, there is an impact on cash flows of both buyer and seller. The cash flows are often small and more stable rather than large irregular payments (Choudhary, 2007a). With the SaaS model customers can estimate their costs much more accurately beforehand than what has been possible in the traditional software business model. This way the SaaS concept can reduce the execution risk for the customers and allow the customers to exit from poor investments with a smaller loss. SaaS vendors are also expected to develop their solutions further and to add innovative product features continuously and this is seen typically as one of the key advantages of the SaaS model.

When selling traditional software products, the buyer is typically the IT department rather than the actual user of the product. When using the SaaS model the situation is often reversed and the buyer is the user (SIIA, 2007). Web-based software solutions have given the users the power to decide what software solutions they want to use regardless of what their company's IT department says. As previously most of the software application sale was focused mainly towards IT decision makers, today software vendors are increasingly targeting business managers and instead of proving how technically advanced their software solutions they want to prove the decision makers of how the solution increases efficiency, decreases costs and so forth.

As the marginal cost of copying software just like any other digital product is essentially zero, the price of the software can fall and does fall to zero as well (Cusumano, 2007). Customers know this and have started fighting against high prices for instance by delaying their purchases until the last week of the end of the fiscal years so that they will get large discounts on license fees. The trend towards zero pricing or free software could be devastating for software vendors unless they can figure out how to convert customers to different pricing models and delivery schemes. The SaaS model can be one solution to this situation. Even if a customer is able to find a suitable open-source software application that would perform the same tasks as a commercial SaaS solution, the customer would still need to organize hosting and maintenance for the application. By paying a small monthly fee the customer gets the application and hosting in one package, often with a Service Level Agreement (SLA) covering service levels, speed and availability and problem resolution standards to ensure a mutually acceptable performance (McNabb, 2001).

In the SaaS world there is no media containing the software product that could be illegally copied and therefore no risk of copyright infringements in the traditional sense (SIIA, 2007). As the software solution is fully controlled by the vendor, the probability of legal issues related to licensing of the application and intellectual property rights is smaller. Acceptable price structure is an important factor, as well (McNabb, 2001). As customers are paying not only for the software application itself but also for hosting, maintenance and other services, temptation to use the software illegally is lower.

2.6 The SaaS model and value chains

Traditionally companies have been atomistic actors competing for profits against each other in an impersonal marketplace (Nordström & Sääksjärvi, 2004). This has been more or less the case also among software vendors. If a SaaS vendor wants to create a highly successful business, in most cases it can not do everything by itself. Instead success requires building a network of key partnerships.

Well-known strategic framework introduced by Porter (Porter, 1985) describes five competitive forces that determine attractiveness of an industry:

- Threat of new entrants
- Threat of substitutes
- Bargaining power of buyers
- Bargaining power of suppliers
- Rivalry among the existing competitors

Based on the analysis of these five forces, a company must decide the type of competitive strategy it wants to adopt (Smith & Rupp, 2002). Generic strategies are cost leadership, differentiation and focusing.

According to Porter any company choosing the cost leadership strategy aims to be the lowest cost producer in the industry and thus the company that is able to offer the lowest prices. This may be the result of economies of scale or proprietary technology, for example. In a differentiation strategy the company seeks to be unique in its industry for instance by producing a product uniquely different from its competitors. The differentiation can be based on the product itself, the delivery system, the marketing approach and a broad range of other factors. If a company chooses focusing as its base strategy it means that the company customizes its strategy to serve a narrow segment or segments within an industry. As Porter says, by optimizing its strategy for the target segments, the company tries to achieve a competitive advantage in its target segments even if it does not possess a competitive advantage overall. An example of how Porter's value chain can be used when analyzing the potential of a SaaS solution is shown in figure 3.

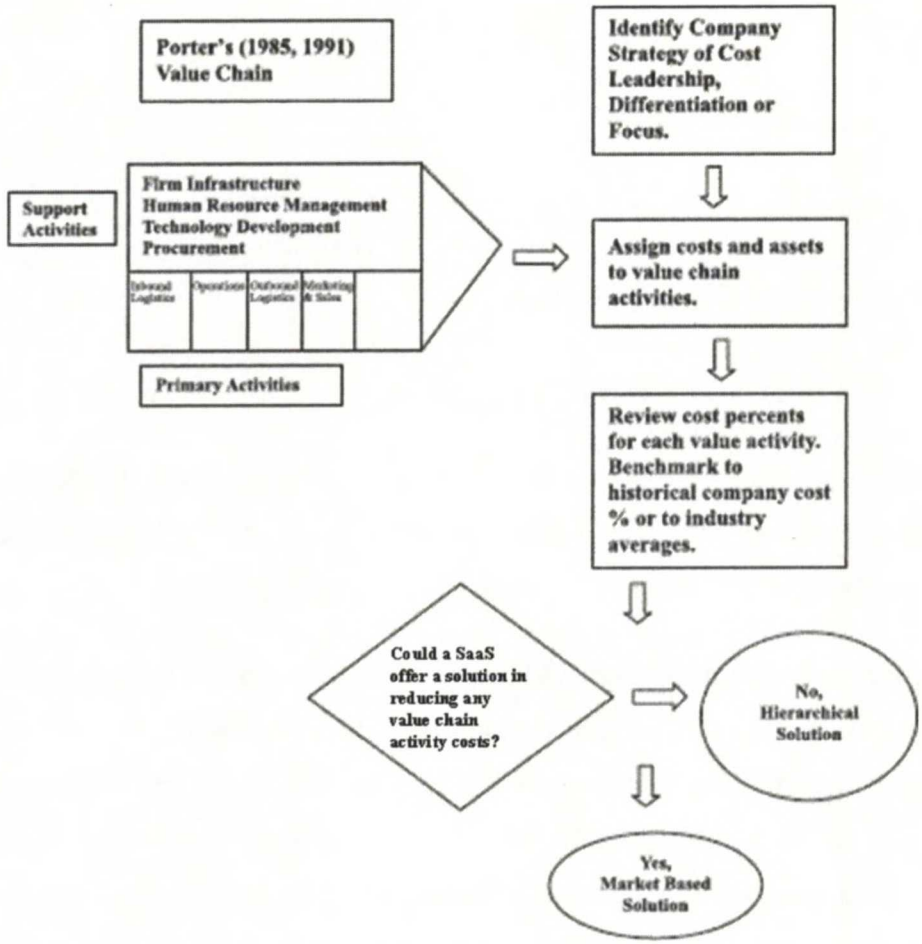

FIGURE 3: Porter's (1985) generic value chain. Source: Adopted from Smith & Rupp, 2002

As with traditional software product business, also companies providing solutions using the SaaS model can rely not only on direct sales but use indirect channels as well. These indirect sales channels are fundamentally different from the indirect sales channels used when selling packaged software products (SIIA, 2007). There is no inventory or logistical issues to be solved, no repair service and no physical goods to be installed or connected.

In a white-paper published by Software & Information Industry Association's (SIIA, 2007) the term 'catalyst' is used instead of the term 'reseller' or 'Value Added Reseller' (VAR) to describe the indirect channels partners in the SaaS world. These partners are more than traditional software product resellers in a sense that they are often parties who understand the business needs of a customer very well. They can, for example, be consulting companies or system integrators working to improve a customer's sales, marketing, HR or logistic processes. In addition, they understand the

advantages and possibilities the SaaS model offers and are able to propose a SaaS model based solution to their customer.

3 Small software vendors' perspective on the SaaS model

3.1 New business models - new business opportunities

Although a SaaS offering is usually delivered over the internet with a business model that is very different from those used traditionally when selling software, it is still about delivering a software application in essence (Sun et al., 2007). The whole SaaS market is yet quite immature and will evolve a lot in the coming years. This opens up many possibilities for new and innovative small software vendors who are familiar with their past and are not afraid of the future.

Earlier software sales have been very much a one-to-one type of business, even when selling software products and not just customized solutions. E.g. a vendor and a customer have negotiated and agreed on the software application that will be delivered, its features and functionalities and also the commercial terms for the deal. Often these negotiations have lasted months or even years. In the SaaS model this has changed drastically and instead of one-to-one business relationships it is a question of one-to-many relationships. Just like the software vendor tries to standardize the application as far as possible, it also should try to standardize the business model and the agreements at least to some extent. In the SaaS model it is not a viable option for the SaaS vendor to discuss and agree of the business terms separately with each customer. Sales cycles can be shortened even to days, which is a great advantage for the vendor.

Software vendors should treat the SaaS model as a strategic area and as an opportunity to increase revenues and profits (Cusumano, 2008). What might seem to be some sort of inflexibility is actually only a way to focus on the core competences and the core business model so that the vendor would be able to serve its customers as well as possible.

The differences between the product and service business are significant and it is not easy to change a company's business model from one to the other (Lassila, 2006). This might also be an opportunity for a new software vendor: If changing an existing business model is difficult for incumbent company, probably starting a fresh company with a fresh business model could be easier. For instance, the SaaS concept is built on a recurring revenue model (SIIA, 2007) and involves small, stable cash flows compared to large, irregular payments under perpetual licensing (Choudhary, 2007a). A healthy financial model is often very different from the traditional software industry and the company must understand concepts such as deferred revenue. This must be taken into account when planning how the company will finance the product development, launching the product, marketing, sales and so forth.

From the vendor's perspective the SaaS model can bring more stable revenue streams and also make customer relationships stronger than compared to the traditional software business models (Rowell, 2007). Still, for a company that has been successful in selling software applications using the traditional models it might be hard to justify the change into the SaaS model even though its importance may be be recognized (Schuller, 2007).

For young startup companies the SaaS model is more of an opportunity than a threat. Web-based solutions have evolved significantly and today there is no reason why the internet could not be used as a platform for business critical solutions (Bennett & Timbrell, 2000). As developing software with web technologies is very cost-efficient, it gives even a small company the possibility to develop a first-class SaaS solution with none or very little external capital. Another big advantage of web technologies is that they make it possible to launch a new software solution to the public without major costs allowing the small company to start selling its product without the need to first invest heavily in building sales channels and sales support organization.

For existing software vendors that are interested in the SaaS model there are several different approaches available (Schuller, 2007). A situation, in which a software vendor decides to create a SaaS offering to replace a soon to be discontinued software product that has earlier been delivered using the traditional methods, is called a full product replacement. Such strategy causes the highest level of cannibalization of the existing market but also might be very rewarding on the long-term, if successful.

Respectively creating a complementary offering is defined as a SaaS offering that is complimentary to a company's existing software product (Schuller, 2007). This kind of strategy might seem lucrative because it does not alienate the existing customer base. Another strategic option is to introduce a light version of an existing product targeting a subset of the customer base. This can be viewed as a poor long-term strategy as it reflects that the SaaS model is merely a marketing tool rather than a new business model that has the full support of the organization.

The SaaS model also alters the relative bargaining power between buyers and sellers as the buyers do not need to make large investments beforehand. This makes the switching away from some solution financially more viable. Then, on the other hand, additional switching costs can incur for example because all application data is stored in the vendor's systems (Choudhary, 2007a). In theory, adding artificial lock-ins to a SaaS solution preventing customers to move away from the application might sound like a good idea but often it is not. The high switching costs might just turn away potential customers who are already concerned about security and reliability of the SaaS model. Instead, an easy and free or low-cost possibility to get one's data back intact potentially increases the attractiveness of a SaaS solution (Ma, 2007).

Even if the internet is a global medium and if internet technologies have been one of the enablers of an accelerated internationalization for many companies, the prediction of the internet as a completely borderless worldwide market place is not realistic (Borsheim & Solberg, 2004). In theory it is true that the market for a new web-based solution is automatically global but in reality it is not the case for many application types. Reasons for this are manifold.

For example many customers want to use a software solution that is available in their local language and therefore the software vendor must be capable of providing localized versions of the solution to be successful. Also local legislation and regulations may cause additional work. For instance, launching a new SaaS solution for financial administration or accounting purposes is not possible if the software has not been localized to fill the specific requirements of each market.

3.2 Sales and pricing strategies

It has been said that the software business is an industry of frequent change and that no changes are more important for both customers and vendors to understand than the changes occurring in pricing (Cusumano, 2007). A decade ago nearly all software vendors, small and large, sold software using only the up-front license fee model (Cusumano, 2008). Since then a lot has changed in the software industry and companies have been innovative in creating new more or less viable sales and pricing strategies.

Any SaaS vendor has a wide range of options for pricing its solution offering. The SaaS model at least partly eliminates the need for manufacturing cardboard boxes, printing manuals, managing stock and so forth (Tao, 2001). Also as there is no software that needs to be installed by the customer, there is no need to provide support for it. As the software application resides on a server, customers can not copy and distribute the application illegally. The SaaS vendor can implement a bug fix and add new features to the product whenever and without the need to inform the customers about it, wait them to download an upgrade patch and install it.

All the above mentioned possibilities for cost savings clearly support the decreasing trend of software pricing that was discussed earlier in this study. This has enabled even very small companies who do not have their own IT departments or any IT skills and who operate with limited budgets to leverage a SaaS offering to run their business better (SIIA, 2007).

For instance, the SaaS vendor can offer a basic version of its solution for free and charge customers only for the use of a more advanced version of the service. Other pricing options include payment on a per-use basis, payment on a subscription basis, payment for a lifetime and so forth. Also, a model in which the solution is namely free but in which the customer needs to pay for some additional features that are actually essential could be chosen.

Despite the large number of different options available and the significance of the decision, the choice between the different models is typically considered to be just a marketing decision that is often done only after the product development (Choudhary,

2007a). Still, according to Choudhary, the business model decision influences a vendor's incentive to invest in product development. In his study Choudhary clearly shows that the SaaS model leads to greater investment in product development and therefore products with better quality under most conditions than the traditional software licensing model.

Under the perpetual licensing model, new features are typically made available as a part of a new product version. In addition vendors also provide so-called patches that typically do not offer substantial new features but fix detected problems and repair features that were promised. The reason for this behavior is that the vendors earn revenues from future product upgrades while these patches are typically provided to the customers for free. Therefore, a profit-maximizing vendor always tries to publish new features as a part of an upgraded and price it accordingly. In the SaaS model all updates are provided to the customers as soon as they become available. This is due to the subscription pricing in which the vendor has an incentive to keep the customers as happy as possible. Therefore, it can be said that in the SaaS model the vendor will typically invest more in software quality compared to a vendor who sells software using the perpetual licensing model.

Many times the product is not sold to a business, but to a group of users. Therefore, marketing of a SaaS offering must also be very different (SIIA, 2007). Adoption of SaaS solutions is partly driven by end-users, who are constantly looking for applications that help them to do their jobs better and who benefit from the possibility to access an application from anywhere and using virtually any web-enable device (Rowell, 2007). This is a fundamental change for many organizations in which IT purchases have previously been led by IT departments and understanding this creates completely new sales opportunities for SaaS vendors. In the best case a vendor is able to make single users to start using its solution and the rest of the company will follow.

While commoditized network and server technologies and widely available open-source software solutions have allowed to launch new companies with less funding than was necessary earlier, the cost of acquiring and supporting SaaS customers is significant (Gardner, 2007). For a small software vendor it is not easy to find companies that are in the middle of making a software investment decision and to persuade them to purchase a SaaS solution instead of a traditional software license.

The key to success are strong partnerships. By building a network of technical consultants and others who often take part in the early phases of a software purchase process, the small SaaS vendor can increase its success-rate significantly. Still, despite of this, most companies offering software products using the SaaS model sell though their internal sales teams (SIIA, 2007).

Success in providing a software product using the SaaS model requires either enabling hybrid revenue logic that allows low priced software offering or customization and lock-in to justify higher prices (Sääksjärvi et al., 2005). According to industry surveys the majority of customers expect that they can integrate the SaaS solution with their existing software applications or other SaaS solutions they use (Sun et al., 2007). This is also an important driving factor to make the SaaS model more attractive.

Today many SaaS vendors deliberately make it as hard as possible for their customers to switch to some other solutions (Ma, 2007). This is done either by introducing long contract periods or high cancellation fees or by using technological measures to lock-in customers. In his study Ma (Ma, 2007) states that this should not be the case in all situations. He says that the contracts should be designed so that there would not be high cancellation fees and that the customers would always have the possibility to get their application data back intact and that the vendor promises to cooperate in the switch. According to Ma, the increased attractiveness of this type of a contract would allow the SaaS vendor to draw users who otherwise might choose the traditional software licensing option. This would then increase the profitability of the SaaS vendor.

The SaaS pricing model typically eliminates separate maintenance payments (Cusumano, 2008). SaaS vendors usually provide their services based on a flat-fee that helps in improving a customer's cost controls. In their study Bennet and Timbrell (Bennett & Timbrell, 2000) present how this can be utilized also in marketing a SaaS offering. First, choosing a SaaS solution instead of a traditional software product often assists the organization in allocating costs in a way that links them more directly to usage. This can then respectively lead to a reduction in excessive user demands, overuse and negligent change requests. Another reason is that the SaaS offering gives the customers more flexibility and allows them to focus on their core competencies. This is important for example when corporate structures are changed for instance

because of downsizing or a merger. It is important also when a customer does not have capital to establish its own IT organization, which is often the case with startup companies, for example.

As mentioned, for companies with limited IT resources such as small and medium sized businesses, the SaaS model is an affordable way to access software applications (Ma, 2007). Still, even though pricing and reduced total cost of ownership from often the spearhead of the argument when selling SaaS solutions, they can not be the only ones. SaaS vendors must offer their customers also strategic benefits such as more secure data, remote access to applications and so fort (Seltsikas & Currie, 2002).

3.3 Gaining and sustaining competitive advantage

Contradictory views exist about whether or not the SaaS model makes it easier or harder to entry the business. According to Sääksjärvi (Sääksjärvi et al., 2005) the entry barrier is relatively high from the perspective of a software application developer. Therefore, the SaaS concept can be seen as a realistic option only if the developer can offer immediate customer value. Some software vendors have been reluctant to invest in the new business model even if their customers have been interested in the SaaS model (Seltsikas & Currie, 2002).

On the other hand, Desai (Desai et al., 2003) sees the SaaS market as highly accessible and thus also highly competitive because it is relatively easy for new entrants to penetrate. What can be said for sure is that from the perspective of a small software product vendor the SaaS model offers new ways to succeed among the competitors regardless of the effect of the model on the entry barrier.

The question is much about which customer segment or segments to target and which types of applications to offer to these target customers. To be successful a SaaS provider must find its own niche. For example, the SaaS model works well for business needs and processes that are being automated for the first time (Levinson, 2007). If there are no legacy processes in place, there is nothing to replace and therefore there are probably fewer change-management challenges and also less resistance towards change. This creates opportunities for innovative startup

companies that can identify such business areas that are not served by the existing vendors and target those niches.

Like in any e-business, the value creation potential of a SaaS offering is based on four independent dimensions: efficiency, complementarities, lock-in and novelty (Sääksjärvi et al., 2005). According to the original definition presented by Amit and Zott (Amit & Zott, 2001) in this context efficiency describes possible reductions in transaction costs. The greater the transaction efficiency gains that are enabled by a particular e-business, the lower the costs and hence of more valuable it will be. Complementarities describe the value potential from bundling products and services together in innovative ways that provide more value than the value of each product or service separately. Lock-in describes the potential value in motivating customers to make repeat purchases. Novelty describes value creation that results from developing more innovative ways to conduct the business. The four value drivers are shown in figure 4.

From the perspective of gaining and sustaining competitive advantage probably the most important value drivers are novelty, lock-in and complementarities. Examples of novelty include launching a SaaS solution offering for some completely new customer segment, for instance developing a small-scale enterprise resource planning solution suitable for small business, or a combining a software application and related services in a new innovative way. Lock-in can also be a positive factor and instead of achieving lock-in through directly or indirectly increasing customer's switching costs the vendor can pursue customer lock-in through different types of loyalty programs and by enabling customers to customize solutions to fit their individual needs and so forth. Complementarities may of course be developed by the solution vendor herself but often they are available also through partnering with third party companies.

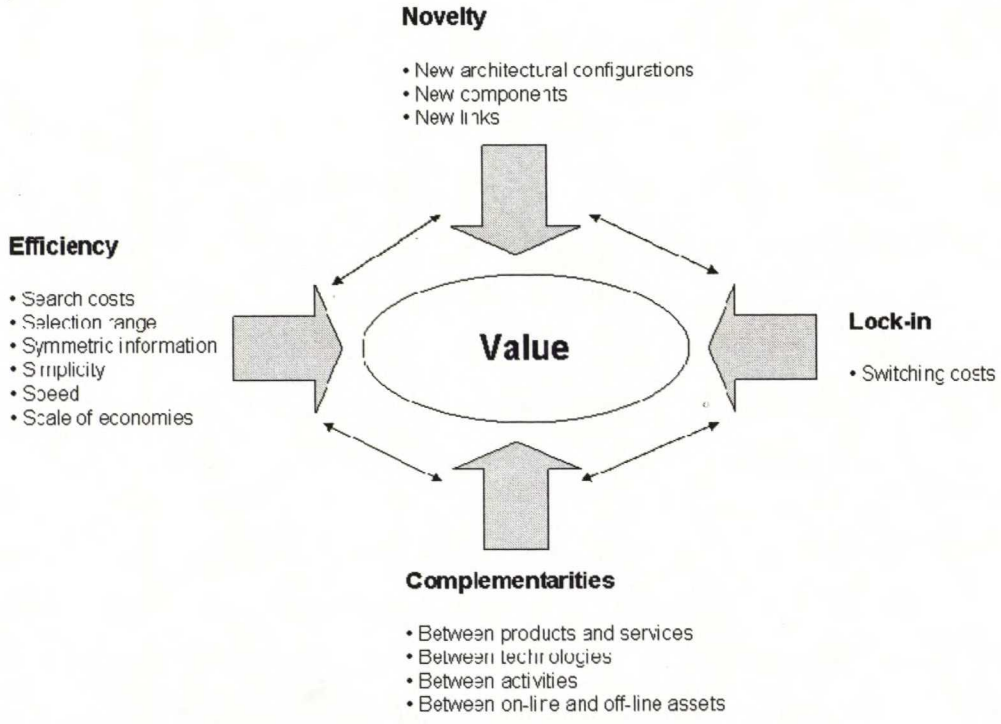

FIGURE 4: The four value drivers. Source: Amit & Zott, 2001

For instance, the customers may need help in maximizing value of their investment and support for business process adjustments. According to the Software & Information Industry Association's white-paper (SIIA, 2007) this is the biggest opportunity for the catalysts. This is especially the case with small and medium sized customers who will still need a catalyst to assist them in making the purchase decision and implementation of the SaaS solution.

Just like the SaaS vendor itself, also its partners must understand the differences between the financial model of the SaaS concept compared to the traditional software product business. A catalyst can make money for instance from referral fees paid by the vendor, by sharing subscriptions revenues with the vendor or by selling additional services such as integration and customization work to customers. The vendor and the catalyst can also create joint operations such as co-operative marketing funds.

Although a software solution is delivered over the internet and does not require anything to be deployed locally, it does not mean that there is no need to train a customer's staff in a new system or new processes (Bennett & Timbrell, 2000). Issues such as communication, change management and training are therefore highly essential parts of a successful SaaS solution sales process. These are exactly such

tasks that can be outsourced to a catalyst. This is especially the case when selling solutions to customers operating in countries in which the SaaS vendor has not established its own presence.

The most important value a catalyst can bring to a SaaS provider is to provide the business process relevant implementation and integration services to the customers (SIIA, 2007). Regardless of the business model chosen by the catalyst, the catalyst must understand the business processes of the customers very well and also have a good understanding of the respective business verticals in order to increase the business of a SaaS provider.

A successful partnership is always based on a joint engagement for a win-win situation (SIIA, 2007). For instance if the SaaS provider views the catalyst just as an external sales force who is even competing with the provider's internal sales team, there is little chance of success. Instead, the vendor should organize training courses for the catalysts, setup innovative and well-structured partner programs to motivate them and so on. In addition, a SaaS vendor must always keep their catalyst partners aware of what they are going to do. If there is a breach of trust, the catalyst can always look for a new company to partner with.

3.4 Technological enablers

It has always been possible to start a new software company with just two individuals working in a garage but today it is easier than ever (Cusumano, 2008). Still even in the late 1990s founders of a new software company had to spend large amounts of money to get such levels of computing and networking capacity and services that are today available almost for free. This combined together with the availability of free open source software has made it viable to launch for example a new SaaS offering with much smaller investment than it was possible before.

The process of network and server technologies getting not only better but also cheaper all the time can be called commoditization (Malik, 2003). In the early 1990s the cost of the server hardware was much higher than it is today. As the prices started gradually decrease started also the adaption of commodity servers for hosting enterprise applications away from mainframe and RISC-based environments (Castro-

Leon et al., 2007). Earlier when a mainframe or a RISC-based system ran out of capacity, it had to be replaced with a new system costing hundreds of thousands or millions. In this situation commodity servers offered an attractive value for money ratio with the option of additional capacity in increments of just thousands or at maximum tens of thousands.

Similarly as the prices for network and server technology have dropped, the advent of free and open source software has driven down software prices and therefore also strengthened the trend of moving from traditional software licensing models to SaaS offerings (Cusumano, 2008). Cheap or free standardized technology platforms reduce the initial infrastructure costs (Nassil & Dasl & Shan, 2007). Today many critical components such as operating systems, databases and application servers are available as free open source software (Cusumano, 2008). A robust SaaS solution can be built using them virtually for free.

Today there is often no need for a new entrepreneur to build a dedicated technical environment as different hosting and network service providers are available for all imaginable needs. Still, the development of new technological and business model innovations has not stopped at these inexpensive hosting service providers. Lately a range of new services have started to emerge under the terms Platform as a Service, Computing as a Service and so forth. All these are very appealing concepts especially for many smaller software vendors: Instead of acquiring hardware and building one's own hosting environment or instead of buying hosting capacity from a third party data center provider, the software vendor can simply specify a required configuration, push a button and a suitable virtual server or data storage spaces are configured automatically. Customers are charged for these virtual servers and storage spaces based on the usage.

Maybe the most well-known examples of the platform as a service concept are computing and storage services offered by the famous e-commerce company Amazon (Amazon, 2008). Its Amazon Elastic Compute Cloud (Amazon EC2) is a web-based service that presents a true virtual computing environment, allowing any software vendor to use web service interfaces to setup a virtual server, load it with vendor's own application environment, manage network's access permissions, and run the software using as many or few virtual servers as needed. Similarly Amazon Simple

Storage Service (Amazon S3) provides a simple web services interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the web.

Also the search engine company Google has recently launched its own platform as a service solution called Google App Engine (Google, 2008). Still in a preview phase, Google's solution offers software developers an easy and cost-efficient way to run their web applications on Google's infrastructure. The goal is to make it easier for web developers to build and scale applications, instead of focusing on system administration and maintenance.

From a SaaS provider's point of view virtual hosting environments like the ones offered by Amazon and Google open up very interesting opportunities. Another significant change in the development practices is the introduction of so-called Service-Oriented Architectures (SOA). In a service-oriented architecture application logic is encapsulated in services that have uniformly defined interfaces and that are made publicly available via different discovery methods (Schroth, 2007). These services may then be retrieved, combined together and used as building blocks for other services.

Software applications are often relatively heterogeneous with respect to technical platforms they are built on, programming languages used and how they can be integrated and customized (Schroth, 2007). Encapsulating them with uniform interfaces enables easier integration between different software solutions and also development of so-called service compositions that are actually new applications built on the basis of existing services. Services offered by Google are a good example of this (Google, 2008). It is unnecessary and inefficient for developers to write components like authentication and email from scratch for each new application. Software vendors who start using Google's Google App Engine solution can make use of its built-in components and Google's broader library of APIs that provide plug-and-play functionality for simple but important features.

A SaaS vendor should try to develop as compatible product as possible (Ma, 2007). In other words the vendor should put emphasis on using open languages and standards, whenever possible, and build application with modular structure and loose coupling. Also, SaaS is likely to strengthen the publisher's incentive to improve the deployment

and maintenance processes (Choudhary, 2007a). In the best case this creates new business opportunities for the vendor as it can start providing other vendors with parts of its own software solution. Just like companies like Google and Amazon have opened their state-of-the-art hosting and software infrastructure for external parties.

3.5 New technological challenges

3.5.1 Security and reliability

Along with a good pricing model, reliability and support services, security is a key to success for any software vendor launching a SaaS offering (McNabb, 2001). This is especially true if the vendor provides its solution to a number of competing customers. As Walsh has put it, an organization relying on an outsourced software solution will take the blame from its clients in the event of system failure (Walsh, 2003). Even a small problem either in security or in reliability can cause customer dissatisfaction that can be difficult to repair. Therefore it is clear that the customer needs deep confidence in the SaaS provider.

A simple example of stock quote service shows well the many difficulties with the security: By using such service the customers makes it known to the service provider that they are interested in a certain stock (Boyens & Günther, 2002). To secure the service, several issues must be addressed that Boyens and Günther cover in their study. First of all, access to the service should be given only to authorized users with the appropriate security level. Also the data connection between the customer and the service must be secure to protect confidentiality and integrity. Last, all user specific data stored remotely at the service provider's infrastructure must be secured so that it is available only to the appropriate user and protected from attacks by any third party.

Typically all business data related to some SaaS solution that a customer is using is centrally stored and managed by the SaaS vendor (Sun et al., 2007). Therefore, securing the data stored in the SaaS provider's environment is one of the key factors in boosting the confidence of potential customers. Horror stories and paranoia about the hacking of business related software systems have made companies skeptical about hosting their business-critical data remotely (Desai et al., 2003). This is especially the case if the service provider is not widely recognized and well-known.

Also, even if there is a relationship of trust between the customer and the vendor, there may be legal reasons why sensate data can not be stored remotely. Examples of such data include for instance medical records (Boyens & Günther, 2002).

As systems become more complex, ensuring their accessibility becomes more important (Brereton, 1999). As the SaaS solutions become more critical for the customers, the SaaS vendor must invest heavily in backup and security systems (McNabb, 2001; Walsh, 2003). Such include backup data systems, computer, power system and network redundancy, automated backup, storage management, data and disaster recovery and multiple contingencies.

Often these security and reliability safeguards go beyond what many small to midsize companies can afford (Walsh, 2003). Still, even if the hosting center managed by the SaaS provider would be more reliable and secure than local servers hosted by the customer itself, network outages can render the whole SaaS application useless. Therefore the network contingency plans must be made very carefully, both by the SaaS application vendor and the customer.

In most cases the customers also require a detailed Service Level Agreement (SLA) to be assured of reliability. Typically these service level agreements include enforcement provisions that give the customer the ability to terminate an agreement and even receive a refund if the provider does not deliver what is promised (Smith & Rupp, 2002). Such provisions include demands for system and data security, continuous system availability and so forth. For example reliability requirements in the range of 99.999 percent are not uncommon (McNabb, 2001). In practice, a demand such as mentioned before means that the SaaS service can be offline only for little more than five minutes per year which essentially means the same as 24/7 availability without any outtakes.

Despite of the SLA's and guarantees given not every customer feels comfortable about leaving sensitive corporate data to the responsibility of a service provider (Boyens & Günther, 2002). This is especially the case if the service provider is not widely recognized and well-known. One method to increase customers' trust on the SaaS offering is to provide them a way to export their data from the system (Levinson, 2007) in case they want to discontinue to use the solution or to migrate to some other

system. Another way to increase trust among the customers is to run a call center that is both accessible and staffed with skillful workforce. Such additional service is an important asset when customers use the SaaS offering as a part of their business critical systems. Also letting third parties like an accounting or a consulting company to audit and certify the SaaS offering can improve the confidence level of the customers and potentially real levels of security and reliability as well.

3.5.2 Performance, customization and integration

Although technological development has created many new opportunities and made it easier to launch a SaaS offering than ever before, it has also brought new technological challenges. For instance, all technology used should be so secure that the SaaS vendor is able to guarantee that the data stored in the service is fully protected and can not be hacked or accessed by outsiders.

Experienced software vendors who are migrating to the SaaS model have often the biggest capability gap in the operational and customer service skills necessary to deliver quality software solutions online (Dubey & Wagle, 2007). To meet the operational challenge of hosting the software solution companies need to develop capabilities to handle large data center operations, systems and network monitoring and so forth. Building a suitable infrastructure for a SaaS offering is a complex task that requires a committed team and a focused effort (Rowell, 2007).

A SaaS vendor must give customers what they are looking for. Availability is one of the important performance characteristics (Walsh, 2003). From a technical point of view ensuring that the SaaS solution is available 24/7 means several new challenges. The vendor must monitor carefully both the application itself and also the hosting environment. The vendor must also beforehand create plans for different kinds of crisis situations, e.g. what happens if a server breaks down, a network outage occurs and so forth. This is especially important if the SaaS provider for instance relies on a third-party that provides hosting services. In addition, the vendor must plan and build the whole environment from the beginning so that it supports load balancing and new capacity can be introduced when needed (McNabb, 2001).

An innovative SaaS provider can also use the monitoring capabilities for its own product development purposes. By monitoring the usage of a SaaS solution the vendor can gain a greater understanding of how its customers interact with the product and discover for example the most and the least popular features and identify which features appear to cause the most problems (Tao, 2001). Such information is very valuable when developing the SaaS solution further.

Similarly like software vendors who want to start to provide SaaS solutions must develop new technical capabilities, they must also change their attitudes toward customer service (Dubey & Wagle, 2007). With multi-tenant software solutions system outages and network problems can affect a large number of customers at once and demand immediate attention. The more mission-critical a software solution is, the more important it is that all problems and failures are handled and fixed without any delays. This means that for example it is not enough to have technical staff available only during the office hours.

Although the multi-tenancy approach brings a number of benefits, it also introduces several additional complexities in application development, deployment and management (Guo et al., 2007). As all customers use the same application for instance, they should not suffer any significant issues in security, performance, availability, manageability or configurability because of the SaaS model. Ensuring this might be especially difficult for software vendors who have not designed their applications for multi-tenancy from the start. Often a massive and time-consuming re-engineering project would be required to change a software application that has not been architected and developed for multi-tenancy to support it.

Still, even developing a multi-tenant software application from scratch involves many new challenges that must be taken into account (Guo et al., 2007). For instance regardless of the size of a customer, each one of them wants to have their specific needs reflected in the software (Bennett & Timbrell, 2000). This is a major challenge for a SaaS vendor as it runs only one instance of the application and as the base product should be kept the same across the customers to gain economies of scale.

A good SaaS solution enables customers to customize their own service in runtime without impacting others (Guo et al., 2007). Traditionally this type of changes would

have required modifications to the program code and application re-deployment. Today it should be possible to perform such operation online and instantly enjoy the benefits of the changes without any cumbersome and long development and deployment projects.

Usually there exists a need to integrate several SaaS services and other software applications together. It is typical that these different services come from different vendors, have various functionalities and use different protocols (Sun et al., 2007). Integrating a SaaS model based application with on-premise IT systems is very much different a task than integrating two local systems (SIIA, 2007). The local system can often be customized as much as is needed to get them to work together. Also it is possible to build different routes for transferring data between the systems, for instance using shared network drives.

When integrating a SaaS solution with another SaaS solution or with some locally hosted software application, multiple different methods exist. Often it is enough just to enable users to log on to the system once and then access all available software solutions using one set of user-ID and password (Sun et al., 2007). User-interface (UI) integration can of course be evolved also much further. Different software applications from different vendors can even be made to look like each other, if necessary. Integration on a program level means that a business process of one software application triggers a business process of another application. Data integration means that the two solutions use a shared database or some other data storage. Different available integration methods are shown in figure 5.

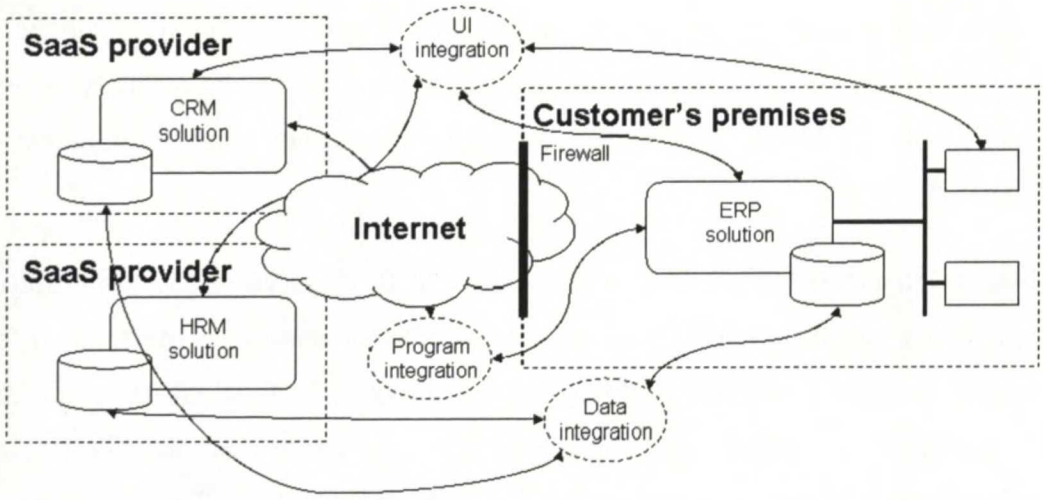

FIGURE 5: Example of SaaS consumption and integration environment Source: Sun et al., 2007

A large proportion of the world's business applications are still so-called legacy programs, e.g. software that has been developed with outdated technology (Sneed, 2006). These are actually the programs that run the information systems of the business world and in many cases migrating to new technologies is impossible without taking these programs along in a way or another. Legacy programs can for instance be wrapped behind different shells and interfaces. Another option is to replace an outdated legacy program with a completely new solution but often that is not a viable option because of for example the costs

At the moment one could say that the legacy problem is likely to get worse. This is caused by the fact that more often the software technology changes the greater is the proportion of legacy software (Sneed, 2006). Also more and more software solutions are built on separate components obtained from many different sources, (Brereton, 1999) which causes more and more integration work.

3.5.3 More rapid development and maintenance cycles

An important factor in the SaaS model is that it is demand-led rather than supply-led (Heide, 2006). It means that the model encompasses the potential for constant change. The SaaS model allows software vendor to introduce more competitive software solutions with shorter life cycles (Aramand, 2007). This is very important in today's competitive market since to survive in the software business software vendors must be

responsive to the changes in the needs and requirements of customers (Greschler & Mangan, 2002; Aramand, 2007).

Software will need to be developed to meet necessary and sufficient requirements (Bennett et al., 2001). This means that for the majority of the customers there is a minimum set of functionalities that the product must support. It is not necessary for the first version of a new SaaS solution to be fully comprehensive and to contain all possible requirements. As new features and functionalities can be added instantly on-the-fly without any action required from customers, the vendor can constantly keep extending the capabilities and functionality of the service to meet the needs of its users.

According to the IEEE standard (IEEE, 2006) software maintenance means the modification of a software product, performed after delivery, to correct discovered problems, to detect and correct latent faults before they are manifested as failures or they become operational faults and to keep a software product usable in a changed or changing environment. Three major classes of the software maintenance work are enhancements, repairs and preventive maintenance (Kemerer & Slaughter, 1997).

Enhancements include adding, changing or deleting functionality of a software product to adapt it to changing business needs. Repairs include corrections to errors found and preventive maintenance includes technical upgrades and restructuring of the software product so that potential future problems are made harmless already before they turn into problems (Kemerer & Slaughter, 1997).

Traditionally software development and maintenance processes have been relatively slow with release intervals of months or even years for very large software applications (Bennett et al., 2001; SIIA, 2007). For most organizations this is far too long as their own business is often dependant on the working software solution and potential problems need to be solved in days or even hours. So-called agile or light-weight software development methods try to answer to this need.

The agile development methods started to evolve in the mid-1990s. The old software development processes were seen as heavyweight, slow, bureaucratic and inconsistent

with the ways how software development was efficiently done in practice. One of the most well-known agile process models is called Extreme Programming (XP).

Extreme Programming and other agile development methods form a framework for software engineering that focuses on iterative software development with short development cycles. It can be said that agile development methods were partly born to satisfy the need of the internet speed development (Heide, 2006). The emphasis is on techniques like dynamic programming, extendable prototyping and unit testing. Constantly ongoing changes are viewed as a natural and desirable aspect of software development. It is more important to be able to adapt to changing requirements at any point during a software development process than to attempt to define all requirements and possible use cases beforehand.

The first two principles of the agile methodologies emphasize the importance of social interactions in the software development process (Heide, 2006). The more the customers can participate in planning and development, the higher is the probability that the final outcome will satisfy the customer. To make the development process faster emphasis is put on working software rather than comprehensive documentation. Responding to changes and not following the original plan strictly ensures that the software adapts to the wishes of the customers. Instead of developing a product that fulfills the original requirements, the vendor should build a product that fulfills the current needs.

Regardless of the software development methods used, software maintenance is a task that is difficult to manage effectively (Kemerer & Slaughter, 1997). Future changes and requirement needs are often difficult to predict. Therefore, it is important that when implementing a SaaS solution, the granularity of the solution is kept in mind (Papazoglou & Yang, 2002). The goal is to minimize the disruption to other parts of the solution if some parts need to be fixed or changed.

Smaller but more frequent software upgrades demand new attitudes on deadlines and quality (Dubey & Wagle, 2007). While schedules for software updates can be flexible, these upgrades need to be virtually error-free. Experience in developing customized software solutions might not be enough to survive in a world like this since often when developing a custom software solution for a customer there are extensive alpha

and beta testing periods, pilot phases and so on before the customer starts to use the new application commercially. Thus, probably companies and teams with experience in developing standardized software products have a slight advantage as they often have the knowhow and skills to implement good-enough quality controls and so on.

Focus on more rapid and better software development and maintenance is not enough but the focus must be expanded from the product and its features to cover the entire chain of actions and all parts of the SaaS model. This includes, for instance, providing customers with online support and suitable documentation (McNabb, 2001). Also from a vendor's own point of view it is crucial that the software solution can be deployed rapidly and that it is easy to add new users, for instance. All these are ways to increase profitability both through keeping costs low and also by reaching more customers as solutions can be taken into production use very quickly.

4 Empirical study

4.1 Research methodology

The empirical part of the research was conducted as a case study. Case studies are one of several ways of doing social science research. Other methods include for instance experiments and surveys (Yin, 2003). In a case study the idea is to research some phenomenon within its real-life context, especially when the boundaries between the phenomenon and the context are not clearly evident and using multiple sources of evidence. The goal is often to find out something new from the topic that has not been found in previous empirical studies (Eriksson & Koistinen, 2005).

Case studies are suitable especially for gaining additional understanding of complex issues and can be used to extend what is already known through previous research. Typically case studies answers to questions like "How" and "Why". As with all research methods, also case studies have been put under criticism. Some of the critics think that a small number of research cases can not offer grounds for reliability or for generalization of results. Some critics even say that case studies should be used only as explanatory tools.

Although qualitative research methods such as case studies originate from sociology and anthropology, they are often used in empirical software engineering research, as well (Hove & Anda, 2005). When the research goals of a study are of a qualitative nature, it is appropriate to rely on qualitative measures. Case studies and for instance interviewing people give insight into their opinions, thoughts and feelings. Still, care is needed when drawing conclusions from a limited number of cases and in ensuring quality of the research (Voss et al., 2002).

Commonly used case study types include for example intrinsic case studies, instrumental and collective case studies, illustrative case studies and exploratory and extensive case studies (Dyer & Wilkins, 1991; Eriksson & Koistinen, 2005). To conduct this case study a mix of the explorative and extensive case study methodologies was chosen. In an explorative case study the aim is to find some new academic knowledge, ideas or theories. In extensive case studies a number of cases are compared simultaneously to find common characteristics and common models.

The case study was prepared and performed according to the following five step process model:

1. Determining and defining the research questions
2. Case selection, determining data gathering and analysis techniques
3. Preparing data collection
4. Collecting the data
5. Evaluating and analyzing the data

The main objective of the case study was to understand what are the opportunities and the challenges of the SaaS model for small software vendors and how the selected case companies see the SaaS model. The research questions were defined to be the following:

- Analyzing the case companies based on the findings from the literature. Do things really work in practice as described in the literature?

- What have been the reasons why the case companies have either decided to use or not use the SaaS model in their business?
- What kind of effects has the SaaS model had on the business of the case companies, e.g. sales, profit and internationalization?
- What challenges and opportunities have the case companies encountered when using the SaaS model?

Data for the case study was gathered from four small Finnish software vendors that develop and sell their own software products to business customers, either as delivered products, as services or as a mix of these two different approaches. The case companies were selected so that they all have a different business focus. Also, it was important to be sure that the companies are willing to participate in the study. Personal relationships with the executive management team members and with the majority owners of the companies were leveraged to ensure the participation.

A web-based questionnaire was sent to 2-4 recipients from each company for collecting general background information. By the deadline eight qualified responses were received, at least one from each company. The data collected from the questionnaire was combined with several informal discussions with representatives of each case company. These discussions did not follow any structured format but instead different topics were often covered in a very detailed way.

The goal of the discussions was to obtain as deep an understanding as possible of the reasons and effects of the SaaS model for each of the case companies. The unstructured approach was selected as the case companies differ quite much from each other in terms of business scope, company age, size of business and so forth. Therefore it would have been relatively difficult to put together an interview structure that would have suited all companies and provided detailed-enough answers of each case company. There was also a risk that some participants might have seen structured or semi-structured interviews as a non-productive activity (Hove & Anda, 2005) and thus would not have been interested in participating in the study. In addition, the unstructured format allowed discovering new topics and ideas that might have not otherwise been considered relevant.

The case study could also have been conducted by using either structured or semi-structured interviews. In structured interviews all questions are standardized and always asked in the same order. The interviewer has specific objectives for the type of information he or she is looking for in the interview (Hove & Anda, 2005). In an extreme case an interview can be so highly structured that all answers can be quantified. Somewhere between structured and unstructured interviews belong so-called semi-structured interviews. Those interviews combine specific questions with open-ended questions. The idea is to bring forth the foreseen information but also to provide the possibility to learn unexpected types of information.

In the following a background description is first given for each of the four case companies. These descriptions cover the companies' history, current product and service portfolios and scope of the business in terms of customer size, internationalization and so forth. After that the SaaS strategy of each company is analyzed separately. The goal has been to understand the reasons and effects of the SaaS model on the business of each individual company. Finally, the findings are analyzed in overall and conclusions are made regarding the SaaS model and the opportunities and challenges it brings to small software vendors.

4.2 Case companies

4.2.1 Overview of Small Planet Ltd

Small Planet Ltd is a Finnish mobile service solution provider that was founded in 1998 as one of the very first companies focusing solely on value added services. It is also the oldest company in this case study.

The company started its operations by developing and selling a multitude of different SMS and WAP based applications enabling its customers to offer consumers mobile event calendars and other information services, mobile games such as quizzes and content download services. Today Small Planet is fully focused on building solutions for mobile marketplace management and mobile social networking for mobile operators and media companies. The company states that its mission is to make mobile entertainment more social and to improve consumer's chances in finding interesting content and people through the mobile media.

At the moment Small Planet employs 12 persons at its Helsinki office. 10 out of those 12 are closely involved with product and service development, including a software development team of four persons, a hosting and support team of two persons and a project and product management team of three persons. Also the company's CEO and COO, who are mainly responsible for sales and marketing, have an active role in the product and service development.

Despite of its small size Small Planet is a relatively international company. During its 10 years of operations it has worked with more than 50 different international customers on five continents and gathered hands-on experience in hundreds of commercial service launches. Currently approximately half of Small Planet's annual revenues come from abroad. The estimated turnover for the year 2008 is approximately 1.5 million Euros.

Today Small Planet has two main products, Download Center CX and Communication Center. Download Center CX is a next generation mobile marketplace management solution, which combines streamlined content management and publishing with a selection of unique social networking features. These include user-created content reviews and automatic content recommendations based on collaborative filtering. Download Center CX is used by mobile operators, media companies and service providers wanting to provide seamless mobile content services for their consumer clients cost efficiently.

The other one of Small Planet's key products, Communication Center, is a mobile social networking solution. It can be used to build mobile chat, dating and social networking services for today's mobile consumers. Mobile operators, media companies and service providers can use Communication Center to create mobile matchmaking services, which allow end-users to set up their virtual profiles, search for interesting users and to chat with them - all in complete anonymity. Small Planet's solution also supports matching profiles based on the geographical proximity of the users, provided that this information is available from the operator network.

In addition to Download Center CX and Communication Center, Small Planet has also its own mobile infrastructure product called MGateway. It hasn't been actively sold lately and is used mainly by Small Planet itself to power its own service offering.

Small Planet delivers both Download Center CX and Communication Center as integrated solutions, installed into the service provider's own hosting environment and managed by the service provider itself, or as a turnkey service. In the latter case the service provider can focus solely on service marketing as Small Planet handles all technical issues. Small Planet also offers complementary services and can, for example, aggregate a selection mobile premium content, including games, video, ring tones, wallpapers and games for its customers. This allows any customer who only has the right marketing channels to start offering its own mobile services.

In addition to its own direct sales resources, Small Planet relies very much on several external sales channels and sales agents, especially in international sales. These companies are either so-called Value Added Resellers (VAR) that have incorporated Small Planet's products in their own product portfolio or simply representatives that act as a catalyst in the deal making process.

4.2.2 Overview of Steam Communications Ltd

Steam Communications Ltd is a leading provider of hosted mobile services in Finland. The company was founded in 2007 as a result of a merger between two pioneering mobile companies, FlyerOne Ltd and Funvision Ltd. In that sense the company is the youngest in this case study but as the both merged entities have been operating already since early 2000s, the company is actually the second oldest. Steam Communications is also the largest company in this study based both on annual turnover and number of employees.

Today Steam Communications' service offering includes mobile marketing, mobile messaging, mobile ticketing and so-called Interactive Voice Response (IVR) services. In addition to services, the company offers also a comprehensive portfolio of products related to mobile devices.

Examples of Steam Communications' products include Steam Engine and Steam AdGate. Steam Engine enables media companies, broadcasters, mobile operators, retailers and other marketing organizations to create and manage a comprehensive suite of turnkey mobile services. Steam AdGate is a professional and versatile tool for service providers for adding and managing mobile marketing messages in existing

mobile services. Steam AdGate controls the logic of the marketing messages and delivers messages to handsets whenever wanted.

Two other products offered by Steam Communications are Steam mCredit and Steam Vote. Steam mCredit is a sophisticated fully automated system for managing so-called immediate short-term loans which are applied for with SMS messages. The system controls the whole loan management chain from handling a loan request to paying out the loan and invoicing the customer. Steam Vote is a voting management system. The system gives the consumer a possibility to participate in polls using multiple different channels including SMS, WAP, web and IVR.

For most of its customers Steam Communications provides the products using the service model: The company takes care of hosting and managing the software solutions. In the past the company has made some solution deliveries when a product has been deployed locally on a customer's IT environment but today even the biggest clients are more interested in turn-key services. Steam Communications did not want to reveal their turnover estimation for the year 2008 but their revenues are mostly generated from sales to Finnish customers. Still, the company is constantly looking for new opportunities to expand its operations abroad.

4.2.3 Overview of Roottori Ltd

Roottori Ltd was founded in 2003 to leverage the opportunity of combining mobile messaging operations of several Finnish companies to one entity so that it would be possible to achieve better economics of scale. It is the second youngest company in this case study and probably the most rapidly growing.

Today Roottori provides its customers with advanced mobile marketing tools and mobile media space and contacts in addition to mobile messaging services. Company's key product called Moutique - Mobile Boutique is a very advanced mobile advertising platform that offers flexible targeting and efficient management of mobile marketing campaigns.

Moutique is a web-based all-inclusive mobile and email marketing tool for marketing campaign planners, designers and buyers. It enables Roottori's customers to capitalize on the numerous possibilities and benefits of mobile and email marketing in a straight

forward way. Moutique has been developed for businesses ranging from small one-man operations to international corporations and it enables the customers to independently create customized mobile marketing campaigns and integrate them as a part of their media mix. High degree of automation and a practical user interface decrease expenses and required time.

Roottori's mobile messaging services are aimed for businesses of all sizes that need the ability to send and receive SMS and MMS messages between their IT systems and consumers' mobile phones. The service is suitable both for bulk messaging needs as pushing marketing and alert messages to a large group of recipients and also for interactive two-way messaging such as mobile value added services.

Since its establishment Roottori has grown very rapidly. Just in a little over a year the company has evolved from a one man operation to a business with some 10 employees. The company aims to achieve a turnover of approximately 2.5 million Euros in 2008. Some 10% of the revenues are already coming from abroad and Roottori is constantly searching for new possible ways to expand its business scope outside Finland.

In addition to building its own base business Roottori has also acquired ownership stakes in several other companies that either leverage its mobile messaging service offering and / or supplement its mobile marketing business.

4.2.4 Overview of Suomen Economia Oy

Suomen Economia Oy is the second youngest and the smallest of the case companies. It is a software vendor focused on developing a web-based Economia CRM solution that is targeted especially for small and medium-sized companies and accounting firms.

Suomen Economia was founded in 2006 as a spin-off from a local system integrator and IT services company called Alphatech Oy which had earlier developed Economia mainly for its internal use.

Today the main Economia product is in its second generation. The main product is a user-friendly, multilingual and multifunctional web-based software solution for

invoicing, project management and other tasks related to running a successful business. Economia also provides tools for customer and product register management and the whole system is easily integrated to IT systems used by customer's accounting firm.

Economia has been designed to meet the needs of companies of various sizes and business areas but it is especially suitable for small companies lacking IT skills and resources. As a web-based and fully managed solution there is nothing to install or to update in the customers' environment. Still, Economia has been designed to be flexible and easy to customize. That is one of Suomen Economia's key business promises - if Economia does not suit a customer's business out-of-the-box, the solution can be customized to meet the needs.

One of the major differentiating factors between Suomen Economia and other similar solution vendors is that Suomen Economia concentrates heavily on building fully automatic links between Economia and IT systems of its customers' accounting firms. This way it creates significant possibilities for process improvement for its customers as the need of sending accounting materials on paper between the parties becomes eliminated.

Although so far Suomen Economia has succeeded mainly in direct sales, it aims to leverage its relationships with accounting firms for reselling purposes. The goal is to build a network of accounting firm customers who recommend the Economia solution to their customers. In exchange the accounting firms would get a cut of revenues brought in by their clients.

Currently Suomen Economia employs just three persons and in addition several freelancers. The company aims to achieve a turnover of few hundred thousand Euros in 2008.

4.3 SaaS strategies of the case companies

4.3.1 Small Planet Ltd - Internationalization through growing service business

As the oldest company in this case study Small Planet started its operations already in the late 1990s. Back then the company aimed to be a traditional software vendor in the sense that all products the company developed were productized, in other words the products were designed and developed to be deployable also outside Small Planet's internal hosting environment, tested and documented adequately and so on.

Sometimes quite a lot of effort was put into productizing process which apparently was away from developing the products further, in other words from adding new features, making enhancements based on customers' wishes and so forth. Still, it is evident that Small Planet itself benefited of these productizing efforts also in many other ways than just increasing sales. For instance, the easier some product was to install and the better it was documented, the quicker and the more cost effective deploying it was even in Small Planet's own hosting center.

Today software services have a significant role in Small Planet's business. Many customers who are actually buying annual software licenses rely on Small Planet's hosting and service management skills. In addition, the company has an increasing number of customers who do not buy traditional software licenses but SaaS solutions.

Small Planet uses several SaaS solutions itself, as well. For example the company buys its email and group calendar services as a SaaS solution from a third party provider. Also the company's invoicing and accounting application is provided using the SaaS model.

Categorization of Small Planet's sales accurately into traditional software licenses and SaaS solutions is not very easy. In all cases it is not always very clear if some deal should be categorized as a license sale or as a SaaS sale, for example when a customer pays an annual license but buys also hosting and service management from Small Planet. On a practical level this is very close to offering a SaaS solution. This said,

depending on the accounting method, approximately 50- 70% of the company's revenues come from services and the rest from annual or perpetual licenses.

Small Planet also uses a lot of different revenue share and other usage based pricing models. Such models are applied both to services hosted by the company but also to services that are deployed to customers' own hosting environments. In some cases the company has full responsibility of managing a certain service even if it is hosted remotely. Categorization of this type of customer deals is difficult, too.

Representatives of Small Planet share quite unified views of the advantages the SaaS model brings. From a software vendor's point of view the most important factors are the possibility to focus fully on developing excellent software products, as there is no need to put effort into making a software application universally deployable, and also the possibility to update products on-the-fly whenever needed. This can have a significant impact on the amount of customer support required as there is no need to train and assist customers in installing, updating and configuring the products. Also the decreasing documentation requirements are seen as an advantage.

From a business perspective the SaaS model is thought to be a way to introduce new innovative pricing and sales strategies. For instance if service deployment costs can be kept low and if the whole organization is able to operate more rapidly than before, the company can potentially target many new customer segments. Such new customers can for example be companies that have previously been too small for commercially viable business cases. In addition the possibility to closely monitor customers' service usage is an advantage in a fast-paced business environment as the lessons-learned can be used as input for further product development activities.

Probably mainly because of Small Planet's long history as a traditional software vendor, its products are all single-tenant at the moment. A separate installation is needed for each customer. On a longer term the company might put development effort to turn its products at least partly multi-tenant but at the moment there are no definitive plans for that.

Small Planet differs from the other three case companies quite much in regards of hosting. All the other companies have at least partly outsourced hosting and

management of their servers and network connections but Small Planet hosts everything in its own internal data center. The company has every-now-and-then analyzed potential benefits of outsourcing the hosting but so far the costs and potential problems caused by the migration have outnumbered the anticipated benefits. One reason for this might be that Small Planet offers every customer a service level agreement and therefore it wants to keep as much control as possible in its own hands.

When taking Small Planet's long operational history into account, its current approach to selling software products seems very natural. The company has gone through many different phases and as markets in general and customers' wishes have changed, so have the business models used by Small Planet. In a way this long background can be viewed as a problem since for example the company's products have not technically been designed especially keeping the SaaS model in mind. Then, on the other hand, a long business history gives Small Planet significant advantage when competing against younger software vendors.

Small Planet has already partly leveraged the possibilities of providing its customers with SaaS solutions instead of deployable applications but it can probably still achieve a lot more. For example by streamlining its operations so that a new service can be put together for a new customer very quickly and with very small costs, Small Planet can continue to search and target completely new customer groups. This is especially important when continuing expansion to foreign countries: Numerous small local and regional companies operate in their own home markets and going head-to-head with them is probably not going to bring the desired results. Instead, if Small Planet is certain about its ability to provide quality services with very low cost, it can decide to rely much more on revenue share deals than before and potentially achieve many times larger upsides.

4.3.2 Steam Communications Ltd - Managed services as a competitive advantage

Steam Communications' roots can be dated back to early 2000s when the term SaaS had not even been invented yet. Still, the company has offered managed services to its customer ever since they have existed. Today some 90% of Steam Communications' revenues come from service sales and only 10% from selling traditional annual or

perpetual software licenses. Also in those cases the company usually provides its customers with hosting and management services.

Just like all the other case companies, also Steam Communication relies on selected SaaS solutions to run its own business. The company has for instance outsourced its email, work-group calendar and financial systems. In addition the company uses also CRM and ERP solutions that are provided using the SaaS model.

Steam Communications' representatives say that the SaaS model provides them with a great number of advantages: The SaaS concept has opened new sales and partnership opportunities and has allowed the company to sell its products to completely new customer groups and create new pricing models. It also has increased the possibilities of international expansion for the company.

Also from a technical point of view the benefits of the SaaS model are numerous. Probably the biggest single advantage is the ability to update a service whenever needed. For example a bug fix that is urgently needed can be deployed instantly and without the need of contacting and updating each customer's own dedicated installation separately. Also, focusing solely on selling software applications using the SaaS model allows the organization to fully concentrate on developing the solution and its features instead of spending time on planning and making the product installable and deployable to external hosting environments. In the same way the SaaS model has also decreased the amount of documentation required as it is often enough to write documents for internal use only and so documentation process consumes less resources.

Mainly because of Steam Communications' long operating history and as the company does not use only the SaaS model, the pricing options it offers to customers are numerous. A customer can either buy an annual or a perpetual license with or without hosting and management services. Another available option is to pay a fixed monthly or quarterly fee that covers both software license and hosting costs. Also different kind of revenue share deals and other usage based agreements are common, just like for Small Planet.

Also similarly to Small Planet and again mostly because of Steam Communications' long history in the business, its software products are not fully multi-tenant. Still, the situation is a little bit different from Small Planet's case as the company uses both product installations that are shared among many customers and also makes separate installations.

Steam Communications has outsourced hosting of its servers but it still administrates them by itself. In this model the hosting partner is only reliable for providing a reliable and secure data center environment. The selected model has proven to work for the needs of the company. For some of its customers Steam Communications offers separate service level agreements. Typically these customers are mainly large corporations that are aware of the need of such agreements and have especially demanded them.

Definitively one reason behind Steam Communications' great success as a mobile value added solutions developer has been its ability to provide even the most complex mobile services to its customers as managed solutions. This has enabled Steam Communications to work with such customers as brands and marketing agencies that usually are not interested in technology at all. Instead of competing with other mobile service developers for the limited amount of potential mobile operator and mobile service provider customers, Steam Communications has expanded its business by reaching completely new customer segments.

The managed services concept has been effective also when reaching out for very large deals. A good example of such a deal is the outsourcing agreement with YLE (Finnish Broadcasting Company) according to which Steam Communications takes care of developing and hosting all YLE's mobile consumer services.

Another excellent example of possibilities of the SaaS model is the mobile ticketing solution Steam Communications launched in 2007. The solution gives event organizers, ticketing companies and other involved parties an easy and immediately available way to start selling mobile tickets for concerts, sports events and even for movie theaters, for example. A consumer can buy a ticket either online or from a ticket shop and the ticket is then delivered to her phone as a mobile message. The ticket contains a machine readable bar code that can be read either with a hand-held

scanner (for instance in movie theaters) or with a mounted ticket reader (for example in sports arenas).

Mobile ticketing provides significant advantages to all parties. From a consumer's point of view mobile tickets are a hassle-free way to buy tickets to events: If ticket purchase is made online, there is no need to go and pickup tickets from any physical store or wait for them to be mailed via standard post. Also, if a ticket gets lost it can easily be replaced just by sending a new copy of the mobile message to the consumer's phone. From the event organizer's perspective mobile ticketing removes the need to organize printing, sales and accounting of paper tickets, which decreases costs and also opens up possibilities for new pricing strategies. For example concert tickets can be sold with a significant discount during the last 24 hours if it seems evident that the show will not be sold out. For a ticketing company mobile ticketing is a way to open up new sales channels such as online stores and it also enables cost reductions. In addition it can assist in managing the problem of pirated tickets as mobile tickets are unique and linked always to a specific consumer.

Steam Communications' mobile ticketing offering covers the whole solution from end-to-end. This is very important as most of the potential customers for the solution are such that they do not want to own any additional hardware or software themselves or manage it but just want to be ensured that the sales and delivery solutions works and is available 24/7. To provide all this, Steam Communications is in an excellent position as it has knowledge and experience of both deliver and service models. Parts of the mobile ticketing solution are deployed at customer's site (for example ticket readers) but also single sales and a database solution is needed and that is offered to the customers using the SaaS model.

4.3.3 Roottori Ltd - Creating new market and business opportunities with an innovative SaaS service

Roottori's business has been based on a service model straight from the beginning. At first the company acted only as an aggregator for mobile messaging connections. It built bridges between systems of different mobile service providers and mobile operators. By achieving economies of scale and leveraging the excellent contact

network of the company's founder, Roottori was able to start doing significant low-margin high-volume business.

Relatively soon Roottori started to expand its service portfolio with different messaging related services such as real time monitoring and statistics tools. This allowed the company to increase its margins as it was no longer selling just bulk messaging connections but could also bring some additional value to its customers on top of aggressive pricing models.

As a big part of the messaging traffic going back and forth through Roottori's system was already in the beginning mobile marketing related, it did not require much time before the company announced its first mobile marketing tools. Also in this case one goal was to offer the company's customers extra services that would bring in better profits than just the message connection sales and another goal was to expand the operations of the company to reach completely new customer segments. For instance advertising agencies and brands are usually not interested in owning the technology they are using and therefore developing services for the needs of those customer groups created new business instantly.

Roottori is one of the two companies covered in this case study that does business only using the SaaS model. The company does not license its mobile messaging and advertising solutions nor deploy them to its customers' own environments. The solutions are also the only ones that have been designed and built to be multi-tenant from the beginning.

Support for multi-tenancy is a great advantage for Roottori and it enables the company to serve its all customer with just one application installation. It makes for instance introducing bug fixes and new features extremely rapid and practical: All new updates become immediately available for all customers.

Roottori itself uses several SaaS solutions in its own business. The company uses outsourced email and a group calendar, its intranet solution is a SaaS service and so on. Roottori has also outsourced hosting of its servers and networks connections although the company administrates them by itself. The company has service level agreements in place with some of its larger customers. Still, as mobile messaging is in

essence a very quality focused business, Roottori of course tries to serve also its smaller clients as well as possible and offers them the same or almost the same service quality as for the larger customers.

Very similarly to the representatives of Small Planet and Steam Communications, also inside Roottori there are very coherent opinions about the advantages of the SaaS model. For example, from a software vendor's point of view the most important factors are the possibility to focus fully on developing excellent software products as there is no need to put effort into making a software application universally deployable and also the possibility to update products on-the-fly whenever needed. Also the decreasing documentation requirements are seen as an advantage.

The SaaS model has also allowed many new innovative pricing models and Roottori has so far been very proactive in testing and using these. Depending on the service, the company charges its customers either no fixed fees and only transactional fees based on messaging, number of targeted consumers and so on or then some relatively small fixed base fee and a little bit lower transactional payments.

Roottori's spearhead product Moutique is a great example of an innovative service that has become possible thanks to the SaaS model. In essence it is a web-based solution for creating mobile marketing campaigns, targeting consumers and sending out advertising messages. As such a product it could also be a software application that would be installed to customer's own hosting environment. Still, the key features of Moutique would not be possible if the service would not be based on the SaaS model: First of all, Roottori has integrated Moutique with several large direct marketing directories and all contacts of those registers are automatically available to all Moutique users. Similarly Moutique is integrated to several different messaging connections and therefore customers can reach consumers in over 200 countries, if needed.

Although Moutique is an advanced software platform as well, the unique features it offers for the customers are mainly not based on a technological but a business innovation. Roottori has leveraged the connection network and business know-how it has gathered during its operations and has created a unique service offering based on that. The more customers use Moutique, the larger the available direct marketing

databases will grow and the better the fully automatic consumer targeting will work. In this sense Roottori has developed a service that truly generates positive network externalities.

The SaaS model suits Roottori extremely well as the whole organization is very sales and new business oriented. The motto of the company is that whatever the customer's problem is, Roottori can solve it. For this type of operations the SaaS concept is the best possible option. Whenever a customer requires something special, it can be implemented in quite a short time period and it will also be possible to sell the same functionality to other customers, too.

4.3.4 Suomen Economia Oy - Targeting non-users with a problem-free SaaS based solution

Suomen Economia is the smallest of the case companies. As is popular among small startups with limited financial and human resources, the company has outsourced its email and group calendar solutions. Otherwise the company is not using any SaaS solutions in running its own business at the moment.

Similarly to Roottori, Suomen Economia relies solely on the SaaS model. The company charges its customers either based on service usage or based on number of users accessing the service. Some of the customers are invoiced monthly and some quarterly. In addition Suomen Economia offers its customers also customization services and charges either an hourly or a fixed fee for them.

Among the case companies Suomen Economia is probably the most typical example of a young startup company that might not have been able to launch its business using the traditional software license based model. In theory the Economia product could well be a standalone software application that would be deployed for each customer locally. In that case it probably would not differ much from hundreds or thousands of similar competing products available in the Finnish market alone. Therefore it can be said that Economia's true competitive advantage comes from its deploying model, in other words from. it is a managed service that can be accessed and used over the internet.

According to the representatives of Suomen Economia the most important benefits of the SaaS model are very much related to the above. The SaaS model has enabled the small organization to focus fully on developing the solution and its features instead of spending time on planning and making the product installable and deployable to external hosting environments. Also it is important that the service can be updated whenever needed and for example a bug fix that is urgently needed can be deployed instantly and without the need of contacting and updating each customer's own dedicated installation separately.

Although it has been said that launching a new SaaS solution is more capital extensive than launching a traditional software application business in which you can sometimes get first customers to pay for the product development costs, Suomen Economia is a good example of how a small company can leverage open-source software for launching its own business. By selecting the development tools carefully and by taking an extremely customer-oriented perspective on the product development, the company has been able to launch its operations with quite a limited capital budget.

At the moment Economia is a partly multi-tenant solution. Depending on a project and specific needs of a new customer, the company either makes a separate customer specific deployment of the solution or a shared installation is used. There are also ready plans to evolve the multi-tenancy further so that at some point Suomen Economia would be able serve all its customers using one single solution.

Suomen Economia has outsourced hosting and maintenance of its servers and network connections to a third party provider. At the moment the company does not offer its customers any service level agreements. This must probably change in the near- future so that the company is able to secure more customers. Soon some of the new potential customers will most probably start asking for agreements on paper instead of just promises and marketing talk.

Currently most of Suomen Economia's customers are relatively small companies with just few employees and typically without any IT knowledge. For such clients it is important that the software application is constantly available and that it does not cause any additional headaches. Economia fills this requirement. As a web-based solution Economia suits also very well the needs of constantly traveling business

persons and distributed organizations as the solution can be used anytime and anywhere in the world and the customer will always know the status of its business.

Economia is a great example of use of the SaaS model for purposes of getting so-called non-users, in other words customers who are using some alternative method to achieve the same goal that could be achieved by using some software application, to use the solution. In Suomen Economia's case those are for instance the customers who provide their accountants with all invoices and other material traditionally on paper (for instance once a month). To make it easier for a new customer to take the step into the world of Economia, the company provides customers for instance with free of charge consultation about improving business processes. The idea is to get customers to understand the benefits of electronic invoicing, for example, and sell the Economia solution to them on the side.

Suomen Economia has also been quite innovative in finding new business partners and sales channels for its solution. Differently from many of its competitors the company has decided to aim more at different catalysts than actual direct customer relationships. For example one strategy is to involve accounting firms and get them to resell Economia to their clients. Depending on the nature of such partnership, the accounting firm acting as a catalyst will get a new source of revenues as it get a share of revenues generated by Economia.

Reasons for creation of the above kind of business partnership network are not entirely monetary. Suomen Economia has understood that even if the solution would be very easy and convenient to use, always some initial education and support is needed. By partly outsourcing this task to its catalyst partners, the company is able to keep its limited resources more focused on developing the solution further.

4.4 Analysis of the case study findings

Based on the results of the web-based questionnaire and discussions with the representatives of the case companies it can be said as a fact that at least people working in the software industry are very familiar with the term SaaS and they also understand its meaning. Everybody also agreed that the importance of the SaaS model will definitely increase in the future. This was also seen in the answers regarding

customers' known or assumed wishes: Most and in some cases all customers are more interested in buying new software solutions as a service than as a deployable software application.

Although the term SaaS was known, many other terms are used as well. Such terms include application service provisioning, on-demand applications, managed software services and so forth. Also the phrase "turn-key service" is used. Even if all these terms and phrases are relatively close to each other, mixing them and not using one term coherently might in the extreme case have a negative effect on sales as potential customers do not fully understand what a vendor actually is offering.

In addition to selling their own software solutions using the SaaS model, all companies covered in the case study use one or few SaaS solutions in their own business, too. Most typical outsourced solutions include email and group calendar and different messaging tools. Some companies are also using other SaaS services such as financial software packages and CRM and ERP solutions.

According to the opinions of the case companies most of heir customers are interested in the SaaS model especially because of lower initial investments required and lower total cost of ownership. Many customers are either trying to find ways to cut their personnel and other costs or at least to limit the growth of the costs. The customers are also interested in new innovative and more flexible pricing models. For example usage-based pricing models allow them to monitor and assign costs in more detail than has been possible earlier with annual or perpetual software licenses.

Other reasons for the customers to choose a SaaS solution instead of a software application delivered using the traditional model include automatic and regular service upgrades and possibility to access the solution remotely. This seems to be quite logical as the popularity of different handheld devices that can be used to access the internet is constantly growing. Some customers also feel that a SaaS provider is able to offer them better reliability and security than they could achieve if they would host a locally deployed application by themselves.

The pricing models used by the case companies varied depending on the age of company and if it has been selling software applications using the traditional license

model or not. The younger companies that have solely focused on the SaaS model tended to have less pricing options and more focused sales strategies. On the other hand, many of the so-called licensing deals for the older vendors could actually be categorized as SaaS sales because the companies are providing their customers both with software and hosting and maintenance services.

When it was discussed how the SaaS model has affected the pricing, the common view was that it has made pricing and sales strategies more complex. One additional problem was that larger competitors that have been defining the market have already created a general price level for the market and a new entrant cannot change those prices. Also, as the amount of potential options from which to choose has increased, customers are more careful when trying to find the best possible software solution vendor for their specific needs. This has caused that if it earlier was possible to charge some of product development expenses to customers it is much more difficult today.

From a technical perspective just one of the case companies had truly multi-tenant applications, two companies have solutions that are partly multi-tenant and one company deploys its products separately for each customer. All companies have multi-tenancy on their roadmaps but it seems that there is still quite much work to do before everything is truly multi-tenant. In a way this was an expected result: If a company has been developing its software products already for a long-time, multi-tenancy hasn't been a relevant issue earlier and now it is difficult and / or very costly to add support for multi-tenancy on top of existing products.

The only case company that is offering a truly multi-tenant solution has developed its application to support multi-tenancy directly from the beginning. The two other companies offering partly multi-tenant solutions have added support for it to their products afterwards. Efforts put into development of multi-tenancy support has been justified both by technical and commercial reasons. Foremost, the amount of technical administrative work and service management costs are lower as there is no need to maintain one software installation for each customer. Also it is easier to add new features to a solution as there is no need to worry about downward or upward compatibility issues and different software versions in use by different customers. Reasons like this make it easier to guarantee the quality of a solution.

All the case companies shared a relatively same kind of a view regarding the business challenges of the SaaS model. For instance today it is much harder to get customers to pay for product development efforts than it was before. Customers also change solutions and vendors more often than before as they are no longer tied to a specific vendor with large upfront investments. This has made it more difficult to "collect the cream" – in other words to charge more money from customers who could be willing or at least capable of paying more. In general it can be said that customers have become more price-sensitive and knowledgeable about different options available for them.

Similarly assumptions about customers' views on the SaaS model were relatively unified. Probably the biggest single issue is the question if a SaaS provider is reliable or not. If a customer does not have enough confidence in the provider, it often wants to host and manage the software application by itself. This is especially the case if a solution is such that it involves confidential data or if several competing customers are using the same solution. New innovative pricing and sales models are also considered problematic as customers do not always understand them.

Other risks and potential problems for the customers involve also issues and complexity of integrating a SaaS solution other existing IT solutions. This covers also problems related to data incompatibility. There have also been some questions about the performance of SaaS solutions and for example how network latency might affect negatively the usability. This is partly related to generic suspicions of web-based software applications.

From a software vendor's own perspective integrations between separate software applications is one of the biggest issues. Integration projects often take a lot of time and resources but still they almost as often exceed budgets and schedules. If a SaaS vendor provides its customers with integration services for a fixed fee, this could be a significant problem.

The above mentioned is quite an interesting finding as according to the companies customers' interest in integration capabilities varies a lot from solution to solution. For some customers the ability to integrate a SaaS solution with some other software

product or service is a very important issue but then on the other hand some of the customers are not interested in it at all.

All the studied case companies use open-source solutions either in development or hosting of their own software applications. This has for instance led to a situation in which none of the companies reported to rely solely on commercial database products. Also different open-source based software development frameworks are very popular among the case companies.

All the case companies provide their customers at least with free email-based support and most also with free phone-based support during standard office hours. Some of the companies offer also extra services such as a 24/7 phone support for an additional fee as well as service training. Three out of the four companies also have service level agreements in place either with all customers or with some customers. Typically such agreements cover things like availability, response times for different situations and so forth.

5 Discussion and conclusions

5.1 Findings

Although there might be some concerns about the validity and reliability of the results of case studies that are conducted in an unstructured manner, such studies have often led to new and innovative insights, development of new theory and also have had high validity with practitioners (Voss et al., 2002). Those are the ultimate users of any research.

The above said, I believe and hope that this study proves to be useful both for small software vendors who are thinking about changing their business models to SaaS and also for new startup companies who are just defining the outlines for their business operations. As an individual I have benefited from the process of conducting this research as I have been exposed to the insights and true everyday business problems of the case companies. It has been a very interesting and rewarding experience.

In chapters two and three I built an overall picture of the SaaS model, including its definition and history. Based on the literature key factors both driving and limiting the

adoption of the SaaS model were outlined and also the way the SaaS model affects business models and value chains was discussed. Then a more detailed look at the SaaS model was taken especially from the perspective of a small software vendor. I described what kind of opportunities the new business models create and how these may affect sales and pricing strategies. Finally I discussed gaining and sustaining competitive advantage, technological enablers and also new technological challenges. In the fourth chapter I combined the theory with practical experiences learnt from four case companies that all were in different business positions and had taken different approaches regarding the SaaS model.

Probably the biggest finding of this study was to note that although there has not been much discussion on how the SaaS model affects especially small- and medium-sized software vendors the companies are already using the model very actively. It is not exaggeration to say that the smaller companies have actually been the ones who have taken the SaaS model in to real-life use, developed it further and found new innovative ways to conduct the software application business. This is proven by the fact that most of the successful large SaaS providers are companies that almost nobody knew about five years ago.

New web technologies and the process of commoditization of network and server technologies have also had a significant impact on the software business. As developing software with web technologies is very cost efficient, it gives even a small company the possibility to develop a first-class SaaS solution with none or very little external capital. Another big advantage of web technologies is that they make it possible to launch a new software solution to the public without major costs allowing a small company to start selling its offering without the need to first invest heavily in building sales channels and supportive organization.

In general it can be said that customers buying software applications have become more price-sensitive and knowledgeable about different options available for them. Therefore a lot of consideration must be put on pricing models: Should customers be billed monthly or annually, how long contract duration should be, what other possibilities exist to generate revenues and so on. I think that smaller companies and startups hold an advantage also from this perspective: Small software providers often have no existing market positions to cannibalize and thus making more innovative and

radical decisions is easier for them. If some pricing or sales strategy does not seem to work, it can be changed. Respectively, if some new strategy works, the whole business can start growing at an unforeseen pace.

Developing software solutions to be run over the internet creates of course new kind of problems and security threats. A web-based application depends on network connectivity which can fail. For a small software vendor it might be difficult to overcome customers' concerns regarding data security and ownership, loss and service's reliability. Working closely with well-known partners often eases these concerns. This is especially true for small software companies which must create networks and partnerships to be highly successful.

When summarizing the above, it becomes evident that the SaaS model creates opportunities but also brings new challenges for small software vendors. Still, both based on the literature and especially on the real business experiences of four small software solution providers, it is safe to state that the opportunities definitely outnumber the challenges. Any small software vendor should at least assess the SaaS model and see how it would fit to its particular needs and situation.

5.2 Suggestions for further study

Although software solutions offered as services are nothing new and unique, the SaaS concept is still in its epiphany. Therefore it is not difficult to name a number of research topics and areas that should be studied more. Some ideas for further research include field studies about why some companies have chosen the SaaS model and some have not, comparisons between SaaS and non-SaaS vendors and so on. Future research should also be done in a significantly larger scale so that instead of four case companies there would be tens or even hundreds participants.

A more thorough study about the effects of the SaaS model on pricing and sales strategies would probably very useful for both smaller and larger companies. As the case study showed, some of the companies offering SaaS solutions today are actually using quite similar pricing models when selling software products using the traditional licensing model. Is this a wise decision or could those companies potentially achieve

more revenues or better margins by adjusting their pricing model or by introducing some new sales methods?

From a more technical point of view it would be interesting to study how the SaaS model has affected development and other technology-related decisions: Are software vendors putting more effort on building truly multi-tenant versions of their products, are companies using more open-source components in their development and so forth. It would also be interesting to learn more about the platform as a service concept, its take up and what kind of possibilities and challenges are related to it.

This study has been about the SaaS model and small software vendors. As there is not very much SaaS research available at all, it would similarly be good to study the SaaS model from a perspective of bigger companies. In addition, one interesting area of research could be how SaaS affects the business relationships of ICT companies. In other words does the idea of so-called catalyst hold in practice or do partnerships and co-operation agreements take some other forms.

6 References

Amazon, 2008. *Amazon Web Services*, http://www.amazon.com/AWS-home-page-Money/b/ref=sc_iw_1_0?ie=UTF8&node=3435361, April 7, 2008.

Amit, R., Zott, C., 2001. Value creation in E-business. *Strategic Management Journal*, Vol. 22, No. 6-7, pp. 493-520.

Aramand, M., 2007. Software products and services are high tech? New product development strategy for software products and services. *Technovation,* Vol. 28, No. 3, pp. 154-160.

Bennett, C., Timbrell, G.T., 2000. Application service providers: Will they succeed? *Information Systems Frontiers,* Vol. 2, No. 2, pp. 195-211.

Bennett, K. et al., 2001. An architectural model for service-based software with ultra rapid evolution. *Proceedings of the 17th IEEE International Conference on Software Maintenance 2001,* Florence, Italy, pp. 292.

Borsheim, J.H., Solberg, C.A., 2004. The internationalization of born global the internet firms. *13th Nordic Conference on Small Business Research NCSB 2004*, Tromsø, Norway.

Boyens, C., Günther, O., 2002. Trust is not enough: Privacy and security in ASP and web service environments. *Advances in Databases and Information Systems*, 2435, pp. 21-40.

Brereton, P. et al., 1999. The future of software. *Communications of the ACM*, Vol. 42, No. 12, pp. 78-84.

Castro-Leon, E. et al., 2007. Architecting the service oriented data center. *Proceedings of the IEEE International Conference on e-Business Engineering 2007*, Hong Kong, China, pp. 694-700.

Choudhary, V., 2007a. Comparison of software quality under perpetual licensing and software as a service. *Journal of Management Information Systems*, Vol. 24, No. 2, pp. 141-165.

Choudhary, V., 2007b. Software as a service: Implications for investment in software development. *Proceedings of the 40th Annual Hawaii International Conference on System Sciences 2007*, Hawaii, USA.

Cope, J., 2000. Definition for ASPs emerging. *Computerworld*, April 2000.

Cusumano, M.A., 2007. The changing labyrinth of software pricing. *Communications of the ACM*, Vol. 50, No. 7, pp. 19–22.

Cusumano, M.A., 2008. The changing software business: moving from products to services. *IEEE Computer*, Vol. 41, No. 1, pp. 20-27.

Desai, B. et al., 2003. Market entry strategies of application service providers: Identifying strategic differentiation. *Proceedings of the 36th Annual Hawaii International Conference on System Sciences 2003*, Hawaii, USA.

Dubey, A., Wagle, D., 2007. Delivering software as a service. *The McKinsey Quarterly (Web exclusive)*, May 2007.

Dyer, G.W., Wilkins, A.L., 1991. Better stories, not better constructs, to generate better theory: A rejoinder to Eisenhardt. *The Academy of Management Review*, Vol. 16, No. 3, pp. 613-619.

Eriksson, P., Koistinen, K., 2005. *Monenlainen tapaustutkimus*. Kuluttajatutkimuskeskus, 2005.

Finkelstein, A., Kramer, J., 2000. Software engineering: a roadmap. *Proceedings of the Conference on The Future of Software Engineering*, Limerick, Ireland, pp. 3-22.

Gardner, T., 2007. *OpSource Newsletter: Financial Implications of the SaaS Business Model*, http://www.opsource.net/news/newsletter/07.11/article_financial_implications.php, April 6, 2008.

Gartner, 2007. *A press release from Gartner, Inc.: Gartner Says Service Providers Must Prepare Now for the Software as a Service Wave (March 6, 2007)*, http://www.gartner.com/it/page.jsp?id=501991, April 7, 2008.

Google, 2008. *A press release from Google, Inc.: Previewing Google App Engine: Run Your Apps on Google's Infrastructure (April 7, 2008)*, http://www.google.com/intl/en/press/annc/20080407_app_engine.html, April 12, 2008.

Greschler, D., Mangan, T., 2002a. Networking lessons in delivering "software as a service" - Part I. *International Journal of Network Management*, Vol. 12, No. 5, pp. 317-321.

Guo, C.J. et al., 2007. A framework for native multi-tenancy application development and management. *Proceedings of the 9th IEEE International Conference on E-Commerce Technology and the 4th IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services 2007*, Tokyo, Japan, pp. 551-558.

Heide, D., 2006. *Shifting from Software Products to Online Services: Changes in the Software Development Process*, http://www.digigen.nl/~dodo/articles/sp2os.pdf, March 24, 2008.

Hove, S., Anda, B., 2005. Experiences from conducting semi-structured interviews in empirical software engineering research. *Proceedings of the 11th IEEE International Symposium on Software Metrics 2005*, Como, Italy, pp. 10.

IEEE, 2006. *IEEE Standard for Software Engineering - Software Life Cycle Processes - Maintenance (ISO/IEC 14764:2006(E), IEEE 14764-2006).*

Kemerer, C.F, Slaughter, S.A., 1997. Determinants of software maintenance profiles: an empirical investigation. *Software Maintenance: Research and Practice*, Vol. 9, No. 4, pp. 235-251.

Kern et al., 2002. *Netsourcing: Renting Business Applications and Services Over a Network*, Prentice-Hall, New York.

Konary, A.M., 2004. *SaaS and enterprise ASP competitive analysis*. IDC, October 2004.

Lassila, A., 2006. Taking a service-oriented perspective on software business: How to move from product business to online service business. *IADIS International Journal on WWW/The internet,* Vol. 4, No. 1, pp. 70-82.

Lassila, A., 2007. Offering mobile security as a service. *Proceedings of the 40th Annual Hawaii International Conference on System Sciences 2007*, Hawaii, USA.

Levinson, M., 2007. ABC: An introduction to software as a service. *CIO.com*, May 2007.

Luit Infotech, 2008. *Difference between SaaS and ASP*, http://www.luitinfotech.com/downloads/saas-asp-difference.PDF, April 12, 2008.

Ma, D., 2007. The business model of "software-as-a-service". *Proceedings of the IEEE International Conference on Services Computing 2007,* Salt Lake City, USA, pp. 701-702.

Malik, O., 2003. *The rise of the instant company commoditization isn't a curse. For a new wave of entrepreneurs, it's a blessing. Here's how they're using it to make millions.* Business 2.0, December 2003,

http://money.cnn.com/magazines/business2/business2_archive/2003/12/01/354195/index.htm, March 26, 2008.

McNabb, P., 2001. *A Sense of Security for ASPs*, http://www.aspnews.com/analysis/print.php/11274_879781, November 25, 2007.

Nassil, I., Dasl, J., Shan, M.C., 2007. The challenges of application service hosting. *Web Engineering*, Vol. 4607, pp. 545-549.

Nordström, H., Sääksjärvi, M., 2004. Application service provisioning as a strategic network - Evaluation of a failed ASP project. *Building the E-Service Society, Vol.* 146, pp. 171-186.

O'Reilly, T., 2004. *Open source paradigm shift*. O'Reilly, June 2004.

Papazoglou, M.P., Yang, J., 2002. Developing methodology for web services and business processes. *Proceedings of the 3rd International Workshop on Technologies for E-Services 2002,* Hong Kong, China, pp. 54-64.

Porter, M.E., 1985. *The Competitive Advantage: Creating and Sustaining Superior Performance*. Free Press, 1985.

Rowell, J., 2007. *White Paper: A Step-by-Step Guide to Starting Up SaaS Operations*. OpSource, Inc.

Schroth, C., 2007. Web 2.0 versus SOA: Converging concepts enabling seamless cross-organizational collaboration. *Proceedings on the 9th IEEE International Conference on E-Commerce Technology and the 4th IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services 2007*, Tokyo, Japan, pp. 47 - 54.

Schuller, S., 2007. *SaaS Strategies for Existing ISVs*, http://www.saasblogs.com/2007/08/28/saas-strategies-for-existing-isvs/, November 11, 2007.

Seltsikas, P., Currie, W., 2002. Evaluating the application service provider business model: the challenge of integration. *Proceedings of the 35th Annual Hawaii International Conference on System Sciences 2002*, Hawaii, USA, pp. 2801-2809.

SIIA, 2007. Channels for the new SaaS industry. *Software & Information Industry Association*, March 2007.

Smith, A.D., Rupp, W.T., 2002. Application service providers (ASP): moving downstream to enhance competitive advantage. *Information Management & Computer Security*, Vol. 10, No. 2, pp. 64-72.

Sneed, H.M., 2006. Integrating legacy software into a service oriented architecture. *Proceedings of the 10th European Conference on Software Maintenance and Reengineering 2006*, Bari, Italy, pp. 3-14.

Sun, W. et al., 2007. Software as a service: An integration perspective. *Proceedings of the 5th International Conference on Service-Oriented Computing 2007*, Vienna, Austria, pp. 558-569.

Sääksjärvi, M. et al., 2005. Evaluating the software as a service business model: From CPU time-sharing to online innovation sharing. *Proceedings of the IADIS International Conference e-Society 2005*, Qawra, Malta, pp. 177-186.

Tao, L., 2001. Shifting paradigms with the application service provider model. *IEEE Computer*, Vol. 34, No. 10, pp. 32-39.

Voas, J., Laplante, P., 2007. The services paradigm: Who can you trust? *IT Professional*, Vol. 9, No. 3, pp. 58-61.

Voss, C. et al., 2002. Case research in operations management. *International Journal of Operations & Production Management*, Vol. 22, No. 2, pp. 195-219.

Walsh, K., 2003. Analyzing the application ASP concept: technologies, economies, and strategies. *Communications of the ACM*, Vol. 46, No. 8, pp. 103-107.

Waters, B., 2005. Software as a service - A look at the customer benefits. *Journal of Digital Asset Management*, Vol. 1, No. 1, pp. 32-39.

Yin, R.K., 2003. *Case Study Research: Design and Methods (3rd Edition)*. Sage Publications, 2003.